PYT
PROGRA
A
Machine

Contents

Introduction
Characteristics of python programming
History of Python language
What is Python?
The advantages of Python
The disadvantages of Python language
The design limitations
Common Terms You Should Know with Python
Python Language Structure
Getting Started with Python
Learning the Basics of Python Programming
Flow of Control
Comments
Escape sequences
Variables and What They Do in Python
Understanding the scope of a variable
Modifying values
Python programming 2
How does Python Execute a Program
Survey of a Simple Program
Some of the Basic Commands, Variables, Statements, and Other Things that you can do
Allocating values to your variables
Multiple assignments
Keywords
Operands and operators
Understanding the decision control structure
Loop Control Statements
Functions
Lifetime and scope variables
List comprehension
What is machine learning?
Sorts of Systems of Machine Learning
What is Supervised Machine Learning?
Classification:
Unsupervised Learning
Association
Semi-Supervised Machine Learning
System Testing
System classification

Line Plot
Images
How to train a model
Gradient descent
Batch gradient descent
Stochastic gradient descent
Mini-batch gradient descent
Regularized Linear models
Mass function of a binomial distribution:

Introduction

Most computer language seems terrible to programmers that are yet to start coding, you might have taken a look at a couple of the popular programming languages, for example, C++ or Java and been somewhat humiliated by what you saw. The pages may have been filled to the edge with letters and images that you simply didn't comprehend, and you got disappointed and just needed to leave. Numerous individuals are scared of programming and feel like it is just unreasonably hard for them. In any case, with the Python programming language which is known as home for all, as it is recommended by programmers around the globe as a good language for beginners to learn, you will find out that it doesn't take much for one to read and understand programming. This book is going to give you the fundamentals that you need to acquire for a smooth Python programming journey.

The world of computer has acquired various kinds of people. Some are keen on bringing in cash by creating their own programs to offer to other people. Why few do play and learn various things about how the system will work. Also, many have dedicated their lives to programming, making it the item that brings home their salary every month whether they work at fixing systems, work in a

company to protect the computer, or doing some other types of computer technology. With regards to computer technology, nothing will be straightforward. Previously you can even get a program to chip away at the computer; it needs to get the right code to make it work. There are a few choices for program creation that a programmer may choose including Java, C++ and Python. Here we will investigate some things about Python and why it is regularly favored over the other programming alternatives. Before you can begin using Python to assume control over your programming needs, it is essential to begin studying it and the whole incredible advantages you will get when utilizing this program. Python is an elevated level programming device, which implies that it is always simple to utilize and peruse, even as a learner. The way of thinking behind the code is lucidness and it has a kind of sentence structure that permits the developer to communicate their ideas without having pages of code alongside it. Readability and simplicity are python's top priorities, which is an extraordinary language for amateurs to begin on the grounds that they will really have the option to read and comprehend the code they are placing in. With different alternatives to programming, they may need to invest a great deal of effort trying to get the code just right, including numerous different objects to get it to work. Be that as it may, with Python, it is kept simple and you will find that it is easy to understand what you are doing.

Characteristics of python programming

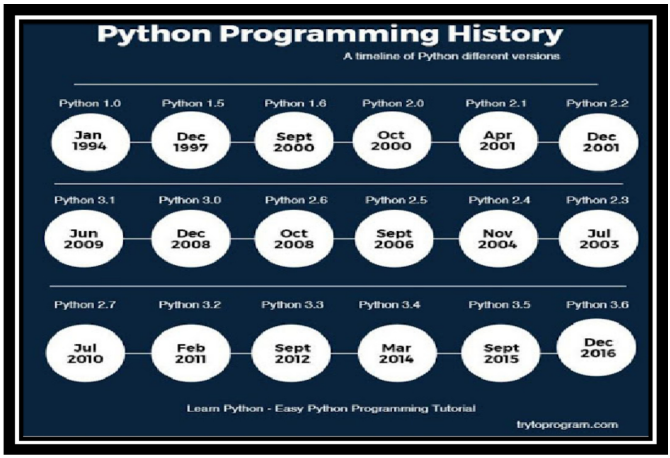
In computer world there are many coding programs that are there for you to use but amongst them python stands tall, this is because python as a coding program has some unique features that distinguished it from other programs. Below are the characteristics of python

1. Well designed syntax: python has a rich grammar which will make the projects so natural to peruse, the language is simply English which makes it work without a great deal of bugs. Python has as a huge library that will work with other programming tasks, for example, evolving documents, scanning for text, and connecting with web servers.
2. Python is extremely intuitive. This makes it simpler for

you to try out little bits of code to check whether they work. You can likewise organize it with a development environment called IDLE. On the off chance that you might want to extend the programming language, it is extend into different modules like C or C++.

3. Availability: Python programming can be run on any unit including Unix, Linux, Windows, and Mac OS X.
4. Programming for all: Python is free. You won't need to pay anything to download and install Python in any device. You can likewise make alterations and redistribute this software. It is under a license, however it is an open source license so others can utilize it as well.
5. Pro features: Despite the fact that Python is a straightforward programming language, it contains some advanced features such as list comprehensions and generators. The software is sensitive to mistakes (errors are easily detected), since instructions are progressively composed, when you combine types that don't coordinate, it will raise an exception for you to take note. You can arrange the codes into bundles and modules if necessary. There is a wide assortment of fundamental data types that you can browse like dictionary, strings, lists and numbers

History of Python language



The Python language was created in 1990 by Guido Von Rossum at Stichting Mathematisch Centrium in the Netherlands. The language itself has been actually developed by a large team of volunteers and is open sourced. At that point, van Rossum was dealing on a project with the Dutch CWI research foundation, that was later closed. Van Rossum used some basics of this new computer language (ABC language) had the option to utilize a

portion of the basics of this new language, known as the ABC language, so as to deal with Python.

In early 2000, Guido and some of the key python development team went beopen.come to form the BeOpen PythonLabs team. In the same year, the PythonLabs team moved to Digital Creations which is now known as Zope Corporation.

Python 2.0 was released on the 16th of October in the year 2000 with features including a garbage collector which helps maintain memory handling related issues in programming. The great thing about Python was that it is backed by a community and it has a transparency behind the users that utilize the Python language.

Python 3.0 which was a major backwards-incompatible version was released on December 3rd 2008 after a long time of testing. The major features of Python 3.0 also have been backported to backwards-compatible Python 2.6 and 2.7.

Python Version Release Dates:

Python 1.0.0 (series) - January 1994- September 5, 2000

Python 2.0.0 (series) - October 16, 2000- July 3, 2010

Python 3.0.0 (series) - December 3, 2008-till date

The fundamental quality of this language is that it is extremely simple to reach out upon to make progressively complex, or keep basic, and it has the option to help various levels of intelligence. Both of these were significant during the days when individual computers were getting into the mainstream. Furthermore, since Python was made to interact with various file formats and libraries, it turned into a hit also. Python has developed a considerable amount since its commencement and more tools have been added to make the programming all the more working. Notwithstanding making Python simple to utilize, van Rossum team has been working at various stages to see that python meets everybody’s need. Using Python language to code can make things simpler and assists with disposing of a portion of the feelings of unease related with the unpredictable computer codes since it doesn't look so alarming. Throughout the years, van Rossum chose to make Python open to all. This permitted all to get entrance and make changes to Python so that if something happened to van Rossum, all would not be lost. On account of having Python open sourced, Python 2.0 was released during 2000 to make it community based and to have a straightforward improvement process. There are a couple more up to date forms of Python 2.0 despite everything being utilized, however Python 3 the latest release stormed the world. Python 3 was released in 2008. It isn't just an update to the program, however a total change. While there are a ton of incredible features

that accompany this version, non is compatible with other so you will need to settle on a decision between Python 2 and Python 3. To make things simpler, the software engineers made a little marker inside the program that would demonstrate a programmer what should have been changed between the two programs when uploading. In spite of this, most have stayed with Python 2.0 for the time being.

What is Python?

Python is generally an object-oriented, high-level programming language with dynamic syntax. Python has been distinguished because of the increased efficiency that it provides since the edit-test-debug cycle is incredibly fast compared to other programming languages.

The Python language has a simple, easy to learn syntax which is empowered with many English words for easier readability and assists for increased productivity and efficiency. When programming in Python, it feels more like you are writing out the solution to a problem in your own thoughts rather than trying to refer to ambiguous symbols that are needed in the language to give to certain functionalities. Python language can be used to automate measurements and process data interactively. Python is able to handle large databases and compute large numbers with much ease compared to many other programming languages. It can be used as an internal scripting language so that it is executed for certain functions by other programs. By learning Python, you will be able to write complex software like intricate web crawlers. It is truly an all-purpose language. The great thing about Python is that it has a giant library for web crawling and is used a lot for its capability in scraping the web. A web crawler is simply a program that can navigate the web depending on the parameters set out for it and scrapes content that you would like for it to scrape. All in all, Python can be easy to pick up whether you're a beginner in programming or an expert on other languages. It's a fast, friendly and easy to learn language but don't mistake that for its powerful nature.

The advantages of Python

Python is presumably a standout amongst other programming languages that you can pick to utilize. Beginners are going to cherish that it is so natural to turn on to this program and begin composing their own codes, even without much experience, and there is bounty to appreciate when you are an expert, or a

specialist, too. Below are some of the benefits one can derive from using python.

1. Simple to use and peruse

With regards to programming language, there are none that are as simple to utilize as Python. Different languages are somewhat cumbersome and difficult to study; you will notice that they have huge amounts of bracket and even words that you can't understand which is sufficient to discourage somebody who isn't used to programming at all since all the words look somewhat scary. Python is somewhat unique. Rather than all the insane brackets, it uses indentations, causing a simpler to peruse page that isn't such a wreck. Rather than words that you can't comprehend, it utilizes English. The other extraordinary characters are kept at minimum with the goal that you can easily study a page of code without any stress. This is one of the simplest programming tools that you can use excel or even to find out more about programming.

1. Effectively present on certain PCs

At times, Python is as of now present on your PC. Macintosh OS X systems just as those with Ubuntu will as of now have Python preloaded. You will just need to download a text interpreter to begin. As far as utilizing Python on Windows PCs, you should simply download the program. Python works with these PCs, regardless of whether it isn't installed from unset.

1. Uses English as its core language

Since English is the language that this program is built on, it is extremely simple to peruse. There aren't a ton of words that you won't get and you won't need to spend the whole day trying to decode the commands. The program is all in English and you will adore how straightforward this can make things.

1. Can work with other programming languages

Before all else, you will probably just utilize Python all alone. It is an extraordinary program to learn with and develop with. In any case, after some time, you may conclude that you need to have a go at something new that Python can't do all alone. Fortunately, Python is ready to work with a few other programming language,

for example, C++ and JavaScript, so you can play around, get familiar with some more, and truly get the code that you are searching for, regardless of whether Python can't accomplish all the work.

1. Can try out things with the interpreter

At the point when you download Python, you will need to download a text interpreter as well. This will make it simpler for Python to peruse your data. You can utilize common products that are already on your PC, for example, Notepad from Windows or search for another interpreter that might be somewhat straightforward. When you select the interpreter that you might want to utilize, the time has come to get to work composing the code. A portion of the individuals who are new to coding may feel stressed about attempting to get the code to work. This is another spot where Python can make things simpler. It will have the option to take the words that you are composing and spit them back, with the assistance of the interpreter, in only a couple of moments. You can test what you are doing while you are chipping away at it! There are such a large number of advantages of utilizing the Python program. beginners are going to love the delightful way promptly accessible this program is and that it is so natural to become familiar with some of the basic commands right away. Indeed, even the individuals who have been programming for some time will be intrigued by how this all functions!

The disadvantages of Python language

While there are a great deal of motivations to adore Python, it is critical to understand that there are a couple of disadvantages in using python that you should look out for. These drawbacks include:

1. It doesn't have a great deal of speed

For the individuals who are hoping to work with a program that has a great deal of speed, Python may not generally be the best choice for you. It is a deciphered language so this will slow it down when compared to different alternatives that are aggregated languages. In any case, it relies upon what you are translating. There are sure

benchmarks with the Python code that can run quicker utilizing PyPy contrasted with different codes. Fortunately this issue with a moderate speed and Python is being cured. Software engineers are trying their best to develop a faster interpreting speed.

1. Not accessible on most mobile browsers

Python is an incredible alternative to utilize in the event that you have a PC. It is accessible on numerous desktop and server platforms to assist you with making the code that you are searching for. Be that as it may, it isn't all set into mobile devices. Since there is such a major increment in income and individuals going into the versatile business, it is tragic that this programming language hasn't stayed aware of the patterns like others. Maybe later on Python will choose to go into the future and build up a software that will have the option to function admirably with different cell phones. Up to that point, software engineers should be happy with utilizing it on their PC and desktops.

The design limitations

On the off chance that you are hoping to work with a program that has a ton of designs the Python program may not be the correct alternative for you. The design language isn't up to what you will discover with other alternative languages. Since you are working with a program that is progressively composed, it takes all the more testing and can have more mistakes that will possibly show up when you are running the program. The worldwide interpreter lock implies that you can just have one thread, access the internals of Python at once. This may not be as significant any longer since it is simple to bring forth the errands out to various procedures, however the plan isn't as decent as other alternatives that you might want. A decent method to work with the task is to recall that space is significant with Python. Other programming languages are going to utilize a ton of bracket to show the distinction in lines and data inside the program, yet Python will depend on spaces. Try to be cautious with utilizing this to maintain a strategic distance from issues and mistakes that can come up.

Python can be probably the best program that you use to compose your own codes furthermore, have a fabulous time. While there are a great deal of advantages to utilizing this program, particularly

contrasted with a portion of different ones that aren't as simple to peruse, it is essential to comprehend both the positives and negatives of every alternative previously you hop in!

Common Terms You Should Know with Python

Before you get excessively far into your programming with Python, it is essential to note a portion of the words that can make the programming simpler to comprehend. This part is going to set aside some effort to take a look at the extraordinary words that are regular in Python programming, and which we do discuss a bit in this book, to help stay away from some disarray and to assist you with beginning with your first code.

Class: this is a format that was utilized for making user defined objects.

Docstring: this is a string that will show up lexically first expression inside a module, function, or class definition. The item will be accessible to documentation tools.

Function: this is a block of code that is conjured when utilizing a calling program. It is best utilized so as to give a calculation or an independent service.

IDE: this represents Integrated Development Environment for Python. This is the essential interpreter and editor environment that

you can utilize alongside Python. It is useful for the individuals who are simply starting up their programming life and can work for those on a careful spending plan. It is just simply a unique code and doesn’t accommodate much space

Immutable: this is an object inside the code that is relegated a fixed value. This could incorporate tuples, strings, and numbers. You can't change the object and you should make another object with an alternate value what's more, store it first. This can be useful at times, for example, the keys in a dictionary.

Interactive: one good thing about Python is that it is so interactive. You can experiment with the interpreter and see how it responds to the new ideals, which is a short way to develop your programming skills.

List: this is a datatype inside Python that is implicit. It contains a mutable sequence of values that are arranged. It can incorporate inmutable values of numbers and strings too.

Mutable: these are the objects that will have the option to change their value inside the program, however which can keep their unique id().

Object: inside the program, this is any information with a state, for example, a value or a quality, just as a characterized conduct, or a strategy.

String—this is one of the most essential sorts that you will discover in Python that will store the content. In Python 2, the strings will store text so that the string type would then be able to be utilized to clutch binary data.

Triple quoted string: this is a string that has three occasions of either the single statement or the double statement. It could have something like '"I love tacos"'. They are utilized for some reasons. They can assist you with having double and single statements in a string and they make it simpler to go over a not many lines of code without issues.

Tuple: this is a datatype that has been incorporated with Python. This datatype is an immutable arranged sequence of values. The sequence is the main part that is immutable. It can contain some variable qualities, for example, having a dictionary inside it, where

the value's can change.

Type: this is a classification of data that is represented in the programming languages. These sorts will vary in their properties, they have mutable and immutable option, just as in their capacities and techniques. Python incorporates a couple of these including dictionary, types, tuple, list, long, integer floating point and string.

In order to install Python on system, you have to download the following:

1 Python IDE

2. Python Interpreter.

The download of these two tools will facilitate the process of becoming a Python programmer. An IDE is a packaged application program used by programmers because it contains necessary tools in order to process and execute code. An IDE contains a code editor, a compiler, a debugger, and a graphical user interface (GUI). There are different types of IDE but the most popular among them is PyCharm for Python. For you to use the PyCharm IDE from JetBrains, you must first install the Python Interpreter (IDLE version 3.4) on your system so PyCharm is able to detect Python on the computer. For you to download the python interpreter use the link bellow: https://www.python.org/downloads/ once you have open the link you will find download then, click download and a file will be downloaded accordingly. Once you have this file, execute it and follow the instructions on the wizard to install the Python Interpreter. After the installation of python interpreter you can proceed to downloading the IDE PyCharm. In order to install PyCharm, visit the link below: https://www.jetbrains.com/pycharm/download/ and click the "Download Community" button and continue to download PyCharm. During this process, it will automatically detect the Python 3.4 IDLE but in rare cases you might need to specify the directory that you installed Python in.

Once you have completed the installation of PyCharm and the Python 3.4 IDLE, go to File New Project. You will reach this screen:

You can rename the "untitled" portion of the Location field to a specific project name. Then click the "Create" button and you will end up being in an empty project. The next thing we must do is create a new Python file which you can do by going to: File New

Python File Set the name of your python file accordingly and it will show under your project name in the Project Explorer. Now double click on the file and you will be met with a page where you can start coding Python.

Python Language Structure

We will learn how Python as a programming language is structured through the following sample code: import math

```
class Program:
  def Execute(self):
      x = math.sqrt(25) print(x)
run = Program()
run.Execute()
```

For you to execute the code, you must use the shortcut alt + shift + f_{10} or execute it using the Run menu item in the main navigation bar in PyCharm. The first line import math is using a unique keyword import which allows the programmer to import tools from the Python library which aren't built-in by default when coding. After the keyword import - the programmer specifies a specific directory that is within the Python library. In this case, we specified the math directory, if we wanted to specifically specify something within the System directory; we would add a dot separator (period/ full stop) and then added a sub directory name. If the library's folders are more complex, you must use notation for instance: import bs4 from BeautifulSoup. The BeautifulSoup library is an example of where we want to import the bs4 directory, which is why the "from" keyword is used in the beginning of the program. In our case, we imported the math module because it contains functions. Many libraries contain functions that are

able to be called so we don't have to code our own functions from scratch. In this case, we used the square root function from the math module. In this case, we used math.sqrt() and then printed out the value it returned relative to the input it received.

We first typed in math to indicate the module we are using and then a dot separator to reference the function within the module which is the sqrt function. The sqrt function takes in a value in its

parameter or parenthesis. While the sqrt function computes the number 25 and returns the value of 5, this value is stored in the x variable which is why it has an equal sign to show that it is equal to the value that is computed from the sqrt function.

Now as we run through each statement, bear in mind that in order to run the next line, we must have a separate line for each statement. The compiler (process to convert language in to readable code that the computer can understand like binary) will then know when statements end and when they start by the use of line breaks.

class - A class can be thought of as a “section” of program. For example: Section: Everything in here is the contents of the section. Again, you will gain a better understanding how classes are useful in Inheritance and Polymorphism.

Program - This second element of this important line is “Program” which simply is a custom name that the user can redefine. You can call this anything, as all it is doing is giving the “Section” or “Class” a name. You can think of it this way: Section Name:

Another thing you must be afraid to look at is the colon: “:” and “}” - what the colon does is simply tells the compiler when a specific section starts. This could be thought of as when someone is writing an essay and they have to start their sentences with the appropriate words or indent their paragraphs. One thing to note is that the spacing in code does matter. If you are creating aclass, the content of that class will be indented once using the tab key in order for it to interpret that the code that the class has authority over is the tabbed code underneath it. Any code that is not tabbed underneath class is not part of that class. This goes the same for any function you create as well. As shown in the above example, a function is defined with a colon and tabbed content underneath it to indicate that it is part of that function. The way Python works is that it has classes, which are sections and functions which are subsections of the class. These classes can be declared as they are declared in the main program after the class is declared and then called with the functions that they carry within them. In this case, it Execute.

Getting Started with Python

Since we know a portion of the advantages of picking this program, the time has come to begin with it. Before you can become familiar with some of the extraordinary procedures that are expected to cause this program to make code for you, the time has come to set up the environment. For the individuals who have a computer with Mac OS X or Ubuntu, you will as of now have Python installed in the system this can make things simpler to begin as you will simply need to click on the symbol to begin. Windows PCs should install Python. While Python works fine and dandy on Windows PCs, it doesn't come preinstalled so you should do this.

These steps works for all Windows version:

Download Python—you can pick between Python 2/ Python 3. Both are awesome alternatives; it just relies upon which one will take care of your business for you. Click to run the Python Installer. At the point when you get to the options, decide to customize Installation. You will see a box pop up. Click on each box that is under Optional features and afterward proceed. On the following screen, search for the Advanced Options and afterward pick where you might want to have Python installed. When you have gotten this far, the following part is to set up your PATH variable. This will permit the user to include directories for all the segments and bundles that are required. To do this progression:

- Open up the Control Panel on your system.
- Look up Environment.
- Under System environment variable, click on Edit. At that point click on environment Variables.

You may need to search a bit for the following part, however search for User Variables. You can then either make another one or alter an existing path. To make another path, select PATH as the name and add it to the directories that are there. Ensure that every Variable Values is separated with a semicolon. In the event that you need to alter your existing path, you have to ensure that each value is on an alternate line. Click in the event that you need to alter your existing path, you have to ensure that each value is on an alternate line. Click on New and afterward put your directories on various lines. Presently you can open your command prompt. To do this click on Start at that point Windows System and afterward

Command Prompt. At the point when the command prompt opens, you can type in "Python." This will load up the Python interpreter. You would then be able to type in Exit and hit Enter to return to the command prompt.

Text editor

You won't have the option to program Python without having the text editor on your Computer. On the off chance that you are utilizing Windows, the Notepad capacity will work. Ensure that you are not utilizing Word because it isn't viewed as an editorial manager furthermore, your code won't save on the system appropriately. In the event that you are considering getting a variant of Notepad, you will see that Notepad + + is the best one to use on a Windows and Text Wrangler is the best to use for Mac. To set them up, fellow the steps bellow:

Windows

- Download and afterward install Notepad + +
- When it is downloaded, open up the settings and click on Language Menu and Tab Settings
- Check the box that is close to Expand Tabs
- Ensure that the value is at 4.
- Click again to Close.

Macintosh

- Download and afterward install Text Wrangler
- You won't have to register to install the product, simply click Cancel if there is a box that pops up requesting this.
- You can as well adhere to the instructions that appears on the screen to set this editor.
- When the program is on your PC the time has come to become familiar with the coding and, functions that you can appreciate on Python.

Getting IDLE

While you are setting up Python, ensure that you download the

integrated Development and Learning Environment. This ought to download alongside Python on the off chance that you are setting it up, however ensure to look into this while you are experiencing the procedure. This is the environment that you are going to work with when you are on Python and it can make things simple. On the off chance that you try not to need to play with finding another environment or you need to make the process as simple as workable for you as a novice, this is the alternative for you.

The primary features of using integrated development learning environment with your Python programming include:

- integrated debugger with continual breakpoints, call stack visibility, and stepping to make things simpler
- Python shell that will highlight the syntax
- Multi-window text editor that can help with the indentation, highlighting and, finishing the code.

Presently, you can decide to utilize another environment, similar to those that we examined above if necessary, however since this one frequently comes as an alternative with Python and it is intended to function admirably with this system, there are numerous individuals who decide to go with this choice. That being stated, there have been a few issues before with IDLE experiencing difficulty focusing, won't copy a few things, and some clients try not to like the interface design. You might need to evaluate this program ahead of time and check whether it is the correct one for you or on the off chance that you might want to utilize one of the alternatives above.

Getting Python set up on your PC is a really simple procedure. There are effectively a few sorts of PCs that have the programming language well present so you won't need to do any work and the rest of them basically need a speedy download to finish. You can stand by only a short measure of time to get Python on your PC and afterward you are a done and can now try out some codes.

Learning the Basics of Python Programming

Presently the time has come to become acquainted with more about Python programming and how you can make it work for you. You should gain proficiency with more about the various watchwords and the factors that accompany Python so you can compose the

code that you need and cause the program to act with a specific goal in mind. Now let's look into some of these essentials of Python programming so you can start creating your new code immediately.

Keywords

At the point when you are chipping away at another PC coding program, you are going to notice that every scripting language will have certain keywords. These are the words that are intended for a particular command in the language and you should ensure not to abuse them elsewhere, in the event that you do (utilize these words in different pieces of your code), you may wind up with an error alert or the program not working appropriately. The keywords that are saved for Python include:

- And
- Pass
- Or not
- Nonlocal
- None
- Lambda
- Is
- In import
- If
- Global
- From
- For
- Finally
- False
- Except
- Else
- Elif
- Del
- Def
- Continue
- Class
- Break
- Assert
- As
- Yield
- With

- While
- Try
- True
- Return
- Raise

Identifier Names

At the point when you are making another program in Python, you are going to work on making many entities, a mix of functions, classes, and variables. All of these will be given a name that is otherwise called an identifier. There are a scarcely any standards that you have to follow while framing an identifier in Python including:

- It ought to contain letters, either capitalized or lowercase or a mix of the two, numbers, and the underscore. You ought not to have any spaces inside.
- The identifier can't begin with a number
- The identifier can't be a keyword and it should exclude one of the keywords inside.

In the event that you defy one of these norms, the program will close on you and will show a Syntax error. Likewise, you have to take a shot at making identifiers that are intelligible to the natural eye. While the identifier may sound good to the PC and get through without causing issues on the PC, a human is the one who will peruse the code to utilize it themselves. On the off chance that the natural eye doesn't comprehend what you are writing in a specific spot, you could run into certain issues. A few of the standards that you ought to follow while making an identifier that is destined to be intelligible to the natural eye include:

- The identifier ought to be spellbinding—you should select name that is going to describe what is inside the variable or will depict what it does.
- You ought to be cautious while using abbreviations that aren't fundamental since these consistently make things that are difficult.

While there are a great deal of ways that you can work out your code, you ought to be cautious and stick with one guideline all

through. For instance, both MyBestFriend also, mybestfriend work in the coding scene, yet pick one that you like and do it the same each time that you work in the program to keep away from disarray. You can additionally include underscores into this or numbers, simply be cautious that you keep things constant.

Flow of Control

When dealing on the Python program, you will work out the statements in a rundown design, much the same as you would when working out a shopping list. The PC will begin with the principal instruction before working through each of them in the order that you make them appear on the list. So you should work out the controls that you need simply like you would for your basic food item shopping list to ensure that the PC is perusing it appropriately. The PC will just quit perusing this rundown once it has done the last instruction to finish. This is known as the flow of control. This is a significant method to begin. You need to ensure that your flow of control is even and smooth for the computer to peruse. This will make it simpler to get the program to do what you want without much difficulty and do ensure that the computer program is free from any issue.

Semi-colons and Indentation

If you examine some other computer languages you will notice that there are lots of brackets used to organize the various blocks of code most times used to start and end statements. This encourages you to make sure to indent the code blocks in these languages to make the code simpler to read, despite the fact that the system can peruse the various codes without any indentation. This sort of coding can make it extremely hard for one peruse. You will see a great deal of superfluous information that is required for the PC to peruse the code, yet can make it hard on the natural eye to understand this. Python utilizes an alternate method of doing this, generally to help make it simpler on the natural eye to peruse what you have. You are going to need to indent the code for this to work. A case of this is:

```
# this function definition starts another block

def add_numbers (a, b):

c = a + b
```

```
# as is this one
return c
# this function definition starts a new block
if it is Saturday
print (It's Tuesday!"
# and this one is outside the block
print ("print this no matter what.")
```

Moreover, there are a ton of languages that will utilize a semicolon to tell when an instruction end. With Python however, you will utilize line ends to tell the system when an instruction will end. You will have the option to utilize a semi-colon if you have a couple of instruction that are on a similar line, yet this is regularly thought about inappropriate behavior within the language.

Letter Case

Most system languages will treat up ercase and lowercase letters equally, yet, Python is one of the main ones that will be case sensitive. This implies that the lower case and upper case letters will be dealt with differently in the system. Remember too that all the reserved words will utilize lower case with the exception of None, False, and valid.

These fundamentals are going to make it simpler to begin on the Python programming. You have to set aside a touch of effort to experience the program all together to get acquainted with it. You won't have to turn into a specialist, yet getting acquainted with some of the text interpreter and few other pieces of the program can make it simpler to utilize and you can figure out how the various buttons will work even before you begin. Evaluate a couple of the examples above first to assist you get started.

The objective of python programming is to keep things simple for the benefit of it users. As should be obvious here, and in the following topics, there are basic commands that you will have to put forward so as to get the program working in a particular way. Study these and you can make an extraordinary program without much stress.

Comments

Python programming language can be used for diverse purposes because it is one of the most intelligent alternatives that you can give a try when you want to code. What's more, since it is so simple to utilize, in this chapter we will go deeper into the study of comments and other unique aspect of python so that you can be able to start creating amazing codes.

In Python programming a comment is one that will begin with the # sign and afterward will proceed until you get to the end of the line. For instance:

This would be a comment

print("Hello, how are you?)

By that comment the system is instructed to print "Hello, how are you?" All comments are ignored in the Python interpreter since it is all the more a footnote in the program to support the programmer, or other people who may utilize the code. They are fundamentally there to state what the program should do and how it will work. It is more descriptive and can be useful without affecting how the code functions. Comment is not ment to be in every line but where needed, on the off chance that the developer feels that something needs clarified better, they would place in a comment however don't hope to see it everywhere. Python doesn't support any comment that will go over a few lines so in the event that you have a more drawn out comment in the program, make sense of how to separate it into various lines with the # sign before each part.

Writing and Reading

A few language programs will usually show the content you need on the screen, or they can demand certain data. You might choose to start programming by explaining the program to the reader, giving it a name or a title can make things simpler so the other coder recognizes what is in the program and can pick the correct one for them.

The most ideal approach to get the correct data to show up is show a string literals that will add the "print" function. Note, string literals that are surrounded by quote which can be a single The sort

of quote that you use won't make any difference that much; in any case, in the event that you utilize one sort to start the statement, you should utilize it till the end. So if there are double quote at the beginning of your expression, ensure you keep to that double quote at the end. At the point when you need the system to show a word or expression on the screen, you would just have "print" and then the expression after it. For instance, if you need to depict "Welcome!" you would do

Print("Welcome!")

With this print, welcome appears on your program for others to use, the print function is going to take up its own line so you will see that after placing this in, the code will consequently put you on new line. Assuming you want the guest to do some other activities with your program you can apply similar principle. For instance, say you need the individual to enter a particular number with the goal that they can pass through the code, you would use the string:

first_number = input('put the first number in')

When utilizing the input feature, you won't consequently observe it print on another line. The content will be put directly after the prompt. You will likewise need to convert the string into a number for the program to work. You mustn't have a particular parameter for this either. In the event that you do the following option with Parenthesis and nothing inside, you will get a similar outcome and most times makes it simpler.

Files

Generally, you will utilize the print function to get a string to print to the screen. This is the default of the print function, yet you can likewise utilize this equivalent function as a decent method to compose something onto a file. A genuine case of this is

With open ('myfile.txt', 'w') as myfile:

Print ("Hello!",file = myfile)

This may resemble a straightforward condition, yet there is a considerable amount of things that are going on in the string over that you should keep an eye out for. In the spot with you opened up the myfile.txt to compose on and then assigned it to the variable called myfile. At that point in the next part, you wrote in Hello! To

the file as new line and then the w told the program that you might have the option to write the changes at the point when the file is open. Obviously, you don't need to utilize the print function to get it to accomplish the work that you need. The write technique will frequently function admirably as well. For instance, you can change the print with compose/write like the instance bellow to get very similar things.

With open('myfile.txt', 'w') as myfile:

myfile.write("Hello!")

So far we have figured out how to print a strings of words into the program and even the most effective method to save them to a particular file. Notwithstanding those options, you can utilize the read method so as to open a particular file and afterward to peruse the information that is there. In the event that you might want to open and read a particular document, utilize this option:

With open('myfile.txt', 'r') as myfile:

data = myfile.read()

with this alternative, you will have the option to inform the program to read the file contents into variable data. This can make it simpler to open up the programs that you might want to peruse.

Built in types

Your system is equipped for preparing a great deal of data including numbers furthermore, characters. The kinds of data that the Python program will utilize are known as types and the language will contain a wide range of types to help make things simpler. A portion of these add string, integers, and floating point numbers. Programmers can even characterize these various types utilizing classes.

Types will comprise of two separate parts. The first part is a domain that will contain a potential arrangement of values and the next part is a set that contains the potential operations. Both of these can be performed on any value. For instance if you have a domain that is a sort of integer, it can just contain integers inside it with addition, division, multiplication, and subtraction. One thing

to note with this is Python is a progressively composed program. This implies that there truly isn't a need to indicate the sorts for the variables when you make it. Similar factors can be utilized to store the values of various types. Regardless of this, Python despite everything needs you to have all the variables with a definitive type. For instance, if the developer attempted to add a number to a string, the Python program would dictate the mistake and show error. It won't attempt to make sense of what you needed; rather it will simply exit.

Integers

On the off chance that you need to utilize integers as a type, you have to keep them as whole numbers. These can be positive or negative numbers, as long as there are no decimals with these numbers. In the event that you have a decimal point in the number, regardless of whether the number is 1.0, you should utilize it as a floating point number. Python can show these whole numbers in the "print" function, yet just in the event that it is the sole argument.

Print (3)

Let's sum two numbers together

Print (1 + 2)

On the off chance that you are utilizing whole numbers, you won't have the option to put the two right close to one another. This is for the most part a direct result of how Python is a typed language furthermore, won't remember them in the event that you join them together. If you might want to put the number and the string together, you have to ensure that the number has transformed into a string.

Operator Precedence

One thing that you have to monitor when you are working in Python is operator precedence. For instance, on the off chance that you have 1 + 2//3 Python could interpret it as (1 + 2)//3 or 1 + (2//3). Python has a technique that will assist you with ordering the operation appropriately with the goal that you get the correct data to appear. For instance, with regards to integer activity, Python deals first with bracket after which other operations that have **, then*, and then//, then %, +, lastly - .

Whenever you are composing an expression that has various operations in it, you will need to remember those signs. This will disclose to Python how to work the numbers so you can find the correct solutions at that point. Remember that most arithmetic operators will be left associative so work it out that way for Python to peruse. The main special case is the ** feature. For instance:

```
# ** is right-associative

2**3**4

# will be assessed option to left:

2**(3**4)
```

Strings

While a string may appear to be something complex, in Python they are fundamentally a sequence of characters. They are going to work a similar way as a list does, yet they will contain more functionality that is explicit to the text. Organizing strings can be a test with regards to composing your code. There are a few messages that won't be fixed string and some of the time there are values that are stored inside variables within it. There is an approach to get this to work directly for string formatting. A case of this is:

```
Name = "Janet"

Age = 24

Print("Hello! My name is %s." % name)

Print("Hello! My name is %s and I am %d years old." % (name, age))
```

The symbols that have a % first are called placeholders. The variables that go into these positions will be set after the % in the order where they are set in the string. In the event that you are doing only a single string, you won't need a wrapper, however if you do have more than one of these, you have to put them into a tuple, with a () enclosing it. The placeholder symbols will begin with various letters, based on the variable type you are utilizing. For instance, the age will be an integer by the name in a string and all these variables will be converted into the string before you can be able to add them into the rest.

Escape sequences

Escape sequence is a way used to denote special characters that are usually difficult to type on your keyboard. What's more, they can be utilized to represent characters that can be saved for something different. For instance, using and in the arrangement can confuse the program so you may utilize the escape sequence to change that for instance:

Print("This is a line. \nThis is another line.")

Triple Quotes

Having dealt with both single and double quotes its important that you understand that at certain time you may need to bring in the triple quote. This is used when you have to characterize a literal that will traverse numerous lines or one that has a great deal of quotes in it. To do this, simply utilize a single and double together or three singles. A similar principle applies with the triple quotes likewise with all the others. You should begin and end the phrase with a similar one.

String Operations

One of the most needful string operations is a concatenation, which is useful while joining a pair of strings together and it’s denoted with the + symbol. There are a great deal of functions that Python can support you with and they will work with the strings to make an assortment of operations. They will have some helpful alternatives that can do significantly more in the Pythons program.

In Python programming, strings are called immutable. This implies that once you make the string, it won’t be changed. You may need to assign a new valuable to a particular variable that exists in the event that you are hoping to make a few changes. There is so much that you can find out about with regards to getting started with Python. It might be a straightforward language, however you need to have the option to figure out how it functions, how to write things appropriately, and even how to leave a comment for others to comprehend when they are glancing through the code. It may appear to be somewhat scary first and foremost, however sooner rather than later, and with some training, you will get it down and be composing your own code within the blink of an eye.

Variables and What They Do in Python

We will be looking straight to the variables; variables are essentially the marks that points to where things can be stored in your system memory and the mostly hold values. At the point when it comes to programming that is typed with statistics, the variables will each have a value that is predetermined and every variable is just going to hold the value of that type. Python has made it somewhat simpler in light of the fact that you can utilize one of your variables so as to store various types. Consider your calculator for instance. The variable will resemble the memory function in this calculating machine. It will hold onto a value with the goal that you can recover it whenever that you need to, however when you store in a more current value, the old ones one will be deleted. The main difference is that you will have the option to have a large number of variables and every one of them will have various values, each of them being alluded by their own name. With Python you will have the option to characterize a variable by giving the mark a value. For instance, you can name a variable count and have it a integer value of one. You would show this by basically writing

```
count = 1
```

Note: that with this synthax, you can allot a value to the variable with the equivalent name. On the off chance that you attempt to access values in a variable that hasn't been characterized, the Python interpreter won't read through this. It will simply exit out of the program what's more, give you an error. You can as well decide to characterize a couple of various variables in a single line, however this isn't the best practice for you to utilize. For instance, you could do this:

```
# Let's characterize three variables simultaneously:

count, result, total = 0, 0, 0
```

Though that is the right method it’s best to display it like this:

```
# This is same as:

Count = 0

Result = 0

Total = 0
```

It is a lot simpler to peruse the subsequent way and will guarantee

that the Python program will comprehend what you need it to state.

Understanding the scope of a variable

You won't have the option to get to each variable from all parts of the program and Not all the variables only one out of every odd variable will be a similar length. The way that you characterized the variable will determine where and to what extent you will have the option to access this variable. The area of your program where you can get to the variable is going to be known as the "scope" and the time that the variable will be accessible is known as the "lifetime". Global variables: are those that are characterized inside the basic file body and you will have the option to see these variables all through the whole file as well as within a file that can import the particular file. These variables have far reaching impacts and along these lines, you may see a few results that you didn't take note. This is the reason a great many people won't utilize global variables, or they will utilize them sparingly. You should possibly include stuff into the global namespace if you intend to utilize them comprehensively, as with functions or classes.

But on the off chance that you characterize a variable within another variable, it will be called a local variable. This one can be accessed from where it is characterized and will possibly exist when that function executes. These are just going to be accessible in specific areas of the program and can't be found or utilized somewhere else.

The assignment operator

We have talked about this option a bit all through the book, yet haven't generally given it a name. The assignment operator is represented by the (=). It is going to be utilized in programming to assign the value to right of the statement to the variable that is found to the left. Sometimes the variable will be made first. In situations where the value on the right is from an expression, for example, a number arithmetic expression, the evaluation will happen before this assignment occurs.

Remember that the (=) won't be a mathematical sign in programming. You can add things to the number and make a wide range of changes that wouldn't make sence on the off chance that you thought of this sign as a mathemtically one. Rather it is an assignment administrator with the goal that the statement will be transformed into the part on the right. At the point when you choose the first value to this variable, you are experiencing the procedure of initializing. The characterization of a value assignment and variable are completed in the single step in this programming, in spite of the fact that it is here and there done in two steps with a portion of the other programming languages. Be that as it may, since it is done in one step, it is more outlandish that the user will commit an error or get an error during the process.

Modifying values

In some programming languages, you will have the option to characterize a unique variable that has a value that has been set. This implies the value can't be changed. These are called constants in the programming language. Generally, Python won't take into consideration these sorts of limitations, yet there is a convention that is utilized to help guarantee that a few factors are set apart to demonstrate that the qualities should be changed. To show this, the names will be written in CAPITAL letters with underscores between each word. An example of a variable that is a constants include:

NUMBER_OF_DAYS_IN_A_WEEK = 7

NUMBER_OF_WEEKS_IN_A_YEAR = 52

Obviously, there are no rules to state you need to put the correct number toward the end. You could state there are 8 days in a week if you want it that way on the grounds that the Python program won't follow along, however it is ideal to simply keep it exact in the event that other coders might want to utilize it. These can be extremely useful to you in your string. Some of the time in the program, for instance, you will need to change the limit of a number that is permitted in the program. This may work fine for a bit, yet perhaps later on you have to increase or reduce this number. Without setting up a constant, you need to experience and make many changes to get everything coordinated. Yet, with a decent constants all together, you can simply return to one spot and get everything repaired. Understanding how the strings work in

your program can have a big effect in the success that you see with this program. You have to realize where they are stored, what the rules are that administer every one of them, and how to make them work in a particular part of the program. With a touch of training, and utilizing the procedures above, you will get this down in no time and can be a veteran as well!

Python programmi 2

How does Python Execute a Program

Each programming language that you work with will execute a program in an unexpected way. This is the reason it is so imperative to figure out how to sort out the words and the various statements that the program requires so you maintain a strategic distance from mistakes and different issues in your code. In this section, we will set aside some effort to figure out how Python will execute the orders that you give and essentially how the entire program runs. At the point when you are working with Python, you are working with a interpreted programming language. You will have a text translator that will execute every one of the programs

going line by line and afterward will change it into a code for the process to comprehend the words what's more, complete them for you. Python is additionally a scripting language, so you can work out the content and afterward save it utilizing the extension.py or you can straightforwardly write it and afterward execute every statement into the Python shell.

Internally, Python is going to work to accumulate your program, fundamentally the source code, into a byte code that has the .pyc extension, much the same as the Java byte code. This makes it simpler for the code to be executed without any delays and you will have the option to see it come up in only a couple of moments rather than waiting around. You will have the option to save your byte code records into a subdirectory that is called pycache which is situated in the directly where the source file dwells. For instance, in the event that you worked out helloworld.py it is going to, at that point be changed over into one of these byte codes and renamed helloworld.pyc. You can go in and manually compile this code if something turns out badly, yet for the most part, Python will do the compilation for you so it won't be an issues. As a starter you may wonder where some of these pyc suffix originate from, yet Python is going to store them with that particular suffixe so whenever it shows up don’t feel threatened. Obviously, this is possibly going to occur if Python has the write access, though even if the python has no write access, it may not be saved that way, but the program will still work. At whatever point you call up a Python program, Python is going to check if there is an existing compiled version with this .pyc suffix. This file ought to be more current than the .py suffix and on the off chance that it exists, the Python will load in the byte code to accelerate how quick the script is capable to go. If the byte code doesn't exist on your system, Python will work to make your byte code before it executes the program. So essentially, each time that you execute a script in Python, you will have a byte code made by the program also. On the off chance that the script in Python is imported like a module, your byte code will be stored in the best possible .pyc file.

Python Implementations

At the point when you hear about implementation of Python, it implies that the program or the environment that is offering support

for executing your programs within the Python language, will be represented with the CPython reference implementation. This implies that it will assist you to work on executing the various codes and statements that you are taking a shot at within the program. There are likewise a few variations of the CPython that you can take a shot at and will have a major effect in the manner that the program works. Some of the features that are availiable with the variations include:

1. CrossTwine Linker—this will be a mix between CPython and an add-on library of your choice. It will offer some better presentation when it goes to the code that you are working on.
2. Stackless Python—this is CPython that has an accentuation on concurency while utilizing channels and tasklets. This is regularly the benevolent that is utilized for the dspython on programs like the Nintendo DS.
3. Wypthon—this is viewed as a re-implementation of some of the parts of Python, which will drop the support of utilizing bytecode so as to utilize the wordcode based omdel. It will utilize the stack register in the implementation and includes in lots of different types of optimization.

implementation is everything with regards to how you can work at your programs and can assist you with getting progressively finished with Python contrasted with some of the other programming languages. What is so exceptional about Python is that it can work with some of other programming languages so as to maintain its versatility and retain the power to get things done easily.Some of the other implementations that you might need to consider on the off chance that you need to accomplish something explicit with the Python programming language include:

- Brython—this is one of the implementations that you can use so as to run Python in your program utilizing a translation to JavaScript so you can use these two together.
- CLPython—this is an implemenation of Python in similar lisp.

- HotPy—this is viewed as a virtual machine for Python that will bolster translation and bytecode improvement.
- IronPython—this is Python in C#. C# is an incredible programming language to utilize within the Windows stages and is frequently a contender to Python dependent on how well known and simple it is to utilize. This implementation permits you to translate your work from Python over to C# in the event that you pick.
- Jython—this is the version of Python accessible for the Java system.
- PyMite—this is Python that you can use for inserted gadgets
- PyPy—this is Python inside Python with the goal that you can focus on a couple of various environments simultaneously.
- RapydScript—this is a language that is like Python that will arrange into
- JavaScript so you can utilize it in the Java stage without going with all of the troublesome language issues.

Taking a shot at Python can be an incredible encounter. On the off chance that you are only a learner with the possibility of programming and are uncertain about how to kick the entirety of this off, some of the other programming dialects can be somewhat confounding. Python is anything but difficult to utilize however has all the power that you need from a portion of the greater names in programming language and you get the advantage of getting the opportunity to utilize this program alongside a portion of the other well known languages that you might need to work with!

Survey of a Simple Program

As we referenced, utilizing Python is one of the easiest programming languages that you can work with. It isn't complex like other languagesIt additionally makes it simpler for you and for another person to go through the information and have the option to peruse it. So how about we take a look at some of the things you can do with Python and how to begin with composing your first program.

The primary program that we will start with is the "Hello World" program. This one is going to require a Python shell to make it

simpler and you will have the option to test it out on your editors on the off chance that you do it appropriately. This makes it simpler to have a smart thought of what you are doing and to detect any errors at the outset. In the event that you are utilizing the Python Shell, which functions admirably on the vast majority of the PC types and programs that you might be utilizing with Python, you will basically need to type in the following programs to get the information to appear:

Print("Hi World!)

You ought to have the option to proceed to execute this data and find that it will appear with the words Hi World! On the screen. This is a basic procedure to do, however it will help you to kick things off and gives a decent review of some basic steps that you need so as to begin composing your own program on Python. Don’t forget that Python is an extremely simple programming language that won't have a ton of various brackets and other data that is standing out that can slow what you are doing. It is additionally extremely simple to peruse.

As should be obvious, you only need a couple of things set up so as to work out the phrase, instead of expecting to compose lines of code to get a similar outcome like you would need to do with other programming languages. We should look into some of the different things that you can do with Python programming and how you can even think of some great code to kick your programs runing.

Some of the Basic Commands, Variables, Statements, and Other Things that you can do

There are such a significant number of things that you can do so as to get a code fully operational on Python. Numerous individuals may abstain from utilizing Python since they feel that it is excessively basic or it simply won't take care of business. However, in realworld, it is basic only for a beginner to learn how to use it. But that does not mean that you wont be able to how tithe way that even a tenderfoot can figure out how to utilize it, yet that doesn't imply that you can't do indept work using python This section is

going to set aside some effort to take a look at the various orders that you can do with Python programming so as to make your projects and codes wake up.

Variables

Variables when mentioned in programming language sound strange to the understanding, but they are simply locations in the memory that are reserved for storing the values of your code. At the point when you take a shot at making a variable, you are saving this spot in the memory. Sometimes, the data type that is in the variable will advise the interpreter to save the memory space and can even choose what you can store on your reserved memory.

Allocating values to your variables

The value will be one of the fundamental things that your program should work with. it very well may be a string, for example, Hi World, 3.14, which is viewed as a type float, or an integer like 1, 2, 3. Python variables won't need an explicit declaration so as to reserve the space in the memory that you need. This is something that will happen consequently at whatever point you place a value with the variable. For this to work, just put the (=) with the goal that the value knows where it is expected to go.

A few instances of this include:

X = 10 #an number assignment

Y= 200 #an integer assignment

Pi (π) = 3.14 #a floating point assignment

Empname = "Arun Baruah" #a string assignment

Remember that when you are working on codes, you can leave a comment with your wok by utilizing the # sign. This permits you to clarify what is happening in the code, leave a few notes, or have something different inside the program. It won't be perused by the translator since it is only a little note that you kept for yourself or for another person comprehension. The following part will rely upon which form of Python you are utilizing. Python 2 is fine with you writing out print and afterward the data you need to discuss but Python 3 will expect you to put the parenthesis in to make it work. For instance:

Print("y = %d" %y)

Print("x = %d" %x)

Print("Employee Name is %s" %empname)

These would then pass through the interpreter and the results you would get ought to be

X = 10

Y = 200

Employee Name is Arun Baruah

Now attempt running this program and observe what comes out. On the off chance that you didn't get the results as listed above you should cross check to confirm were the error is coming from. This is an easier way try out your abilities on python programming.

Multiple assignments

Notwithstanding working with the single variables that were listed above, you will likewise be ready to work on multiple assignments. This implies you will be ready to assign one value to a few distinct variables simultaneously. To do this, you would simply require to put the equivalent (=) sign between every one of them to keep things sorted out and to tell the system that the value will be with the whole of the factors together. You can keep them separated out if that is best for you, however utilizing this technique is going to assist you to send everything to a similar memory location on the system and will give the code a more clear look on your screen. An instance of how to give more than one variable a similar value includes:

a = b = c = 1

From the above you want to let the code know that all the variables are of the same value and that you want all the variables to be at same location within your memory.

Standard data types

In addition to the goodies of python another thing that you're able to work is the various data types which will be used in your code to define the operations that you can do on each data type and explain to others the storage method that will be best for this kind

of data. Python has five standard data types, they are:

Numbers: The Number data types are the ones that will store the numeric values. They will be made as objects once you assign a value to them. There are additionally four unique kinds of numbers that Python will support they are :

- Complex, (for example, complex numbers)
- Float (floating point real values)
- Long (Long integers that can likewise be seen as hexadecimal and octal.)
- Int (signed integers)

One thing to note is that while Python will permit you to utilize the lowercase l while doing the long type of a number, it is ideal to go with a capitalized L at whatever point you are utilizing the letter. This is going to keep you away from confusion while reading the program between the l and, the 1 as they look extremely comparable. Whenever that Python is showing a long integerthat has the l in it, you will see the capitalized L.

Tuples

The tuples share some similarities with the lists but the difference is that tuples uses different signs. The primary difference however is that lists will utilize brackets and the components, just as the size, can be changed through the program. Although, the tuples are going to utilize brackets and you won't have the option to update them. A decent method to consider tuples is that they will resemble a read only page. Once you don't attempt to make changes to the tuple in the program, you will be able to utilize it similarly as you did the list examples above. This makes it a good option to utilize it in case you're interested on something that is straightforward yet won't let anybody make changes to the program after you are finished.

Lists

Lists are one of the most flexible data types that you can take a shot at in Python. In this language, the list will contain various things that are either enclosed with the square brackets or separated out with commas. They are like the clusters that you would find in C on the off chance that you've worked with that program. The one distinction that surfaces with these is that the items that are in a list

can be from various data types. The values that are put away inside the list can be gotten to with a slice operator also as the [:} image with the indexes beginning at 0 toward the start of the list and afterward going down until you get to - 1. The plus sign will be the concatenation operator while you can utilize the asterisk as the repetition operator. For certain instances of what this means and how you can utilize the various signs inside your programming, think about a few of these models:

list = ['mainu', 'shainu', 86, 3.14, 50.2]

tinylist = [123, 'arun']

print)list)#prints complete list

print(list[0]) #prints the first component of the list

print(list[1:3]-#prints components beginning from the second component and to the third

print(list [2:]) #prints the whole of the components of the list beginning with the third component.

Print(tinylist*2) #prints the list twice.

Print(list + tinylist) #prints the concatenated lists.

Strings: Strings are distinguished in Python as a contiguous set of characters that will be displayed by the utilization of quotations marks. Python will permit either double quote or single quote, however you do need to keep things sorted out. This implies in the event that you utilize a double quote toward the start of your string, you have to end that equivalent string with the double quote. The equivalent goes when you are utilizing a single quote. Both of these will mean the same thing, you simply need to ensure that you are utilizing the best possible quote marks to make the code look great and to abstain from making the Python program look complex.

Notwithstanding having the option to print off the string that you might want, you are also allowed to instruct the program to print simply part of the string utilizing some exceptional characters. How about we look at some of the examples of what you can do with the strings, and the corresponding signs that you will use as well, to help illustrate this point.

str = hi Python!'

print(str) #prints complete string

print(str[0]) #prints the first character of the string

print(str[2:5]) #prints characters beginning from the third to the fifth

print(str[2:]) #prints string beginning from the third character

print(str*2) #prints the string two times

print(str + "Guys") #prints concatenated string

Generally you are most likely going to need to print out the whole string to leave a message up on your program so the first print that you do will be sufficient. In any case, if you simply need to print out Hi or some other variation of the words above, you may discover that different options are extremely valuable. You can do any combination of these, they are only guides to assist you as a first timer!

Dictionary

Dictionaries are another sort of hardware that you can utilize when you are working in Python. They are like a hash table type and they are going to work like the hashes or the arrays that you can discover on other programming languages like C# and Perl. They will likewise comprise of key value pairs and keeping in mind that the key can be practically any type on Python, you will see that they are typically going to be strings or numbers. For the most part, with regards to values, you will find that they are an arbitrary objects in python. A few instances of how this will function include the accompanying codes:

#dictionary stores key-value pair, later to be retrieved by the values with the keys

dict = {}

dict['mainu'] = "This is mainu"

dict[10] = 'This is number 10"

empdict = {'name': 'arun', 'code':23, 'dept': 'IT'}

print(dict['mainu']) #this will print the value for the 'mainu' key

print(dict[10]) #this will print the incentive for the 10 key

print(empdict) #this will print the complete dictionary

print(empdict.keys()) #this will print out the whole of the keys

print(empdict.values()) #this will print all the values

One thing to remember is that these dictionary values won't be stored in an organized manner t. They won't have the concept of ordering among the components. This doesn't imply that you can say that the components are out of order, they are simply going to be unordered.

Keywords

The majority of the kinds of programming languges that you will manage will have a few keywords that are saved as a component of the language. These are words that you truly shouldn't use in your code except if you totally can't resist. There are 33 keywords found in the latest version of Python and you should spell them appropriately on the off chance that you need them to carry out the responsibility that you want them do. The 33 keywords that you should keep an eye on include:

False

Class

Finally

Is

Return

None

Continue

For

Lambda

Try

True

Def

From

Nonlocal

While

And

Del

Global

Not

Yield

As

Elif

If

Or

Assert

Else

Import

Pass

Break

Except

In

Raise

Keep this list handy in the event that you are stressed over learning the language. It will have the option to help you out whenever that you have issues with the interpreter about the names that you are giving the variable. You might be confounded regarding why it is giving you a few issues with the words you picked, you can through this list and check whether you utilized one of the keywords inappropriately within your code.

Statements

At the point when you are composing your code in the Python language, you will be making expressions and statements to get the program working. Expressions will be able to process the objects and you will discover them installed within your statements. A statement is essentially a unit of code that will be sent to the interpreter with the goal that it may be executed. There are two

sorts of statements that you can utilize; assignment so far and print. You will have the option to compose the statement, or multiple statements, utilizing the Python Shell to do so intuitively or with the Python script utilizing the .py extension. At the point when you type these statements into the interactive mode, the interpreter will work to execute it, as long as everything is appropriately set up, and afterward you can see the outcomes shown on the screen. When there are many lines that you have to write in code, it is ideal to utilize a script that has an arrangement of statements. For instance

```
#All of these are statements
X = 56
Name = "Mainu"
Z = float(X)
Print(X)
Print(Name)
Print(Z)
```

Operands and operators

There are a ton of incredible symbols that come upwhen you make a code in your Python program. It is imperative to comprehend what parts you can work with and what they stand for. Operators are regularly used to mean subtraction, addition, division, and multiplication. The values of the operator will be called operands. You can utilize various signs for these, so as to get the values that you might want to see.

While you are utilizing the operators and operands, you have to recall that there is

going to be an order of evaluation depending on the rules of precedence. While working on arithmetics Python is going to utilize the abbreviation PEMDAS which is bracket, exponentiation, parenthesis, exponential, multiplication, and division. On the off chance that there are some of these that are the same, for example, two arrangements of numbers that should be multiplied together, you will require to work from left to right to get the accurate

number. Another significant operator that you should search for is the modulus operator. This one is going to work with whole numbers and is going to yield the remainder once the first operand has been divided by the second one.

Note: one other good thing about python is that you can send out, which permits you to explain what you are working on instead of stressing the other programmer that will read the code to comprehend the code that way. You will simply introduce the # sign to denote that a comment/statement about the code was left. The program interpreter whenever it comes across the # sign will ignore it but it won't be removed because another programmer may need to look at the code if need be.

Understanding the decision control structure

Human beings at some times will have to make decisions due to the situation around them, assuming you wake up one morning nd wanted to go for work, but it began to rain out. Did you simply stay there confused without another alternative to support you out? No, you may have woken up and chosen to take an umbrella and go on the walk anyway or you may choose to remain at home and do something else. You can settle on different choices, regardless of whether the first doesn't work out, in view of the conditions around you. Presently this is somewhat a similar thought with regards to working with Python. So far we have just advised the program to do each thing in turn. On the off chance that the circumstances don't arrange precisely with the program, you won't get any outcomes. This won't work for a few programs, particularly if the others are permitted to pick from several unique answers. The decision control structure is the part that permits you to pick a couple of various options for Python on the off chance that the first decision doesn't work out.

Generally, these are going to work at a true or false kind of outcome. You will need to make sense of which action you need to make and what statement the program ought to execute if the result is either true or false. In Python, any answer that is non-null or non-zero will be viewed as true and the ones that are either invalid or zero will be seen as false. For more understanding of these consider

the following: To see a portion of these think about the accompanying:

- If statements: the if statement is going to comprise of a Boolean expression that is at that point followed by at least one statement that will be executed if the appropriate response coordinates.
- If… else statements: this option will have a statement that will appear if the "if" statement is right, however there is likewise another statement that is permitted to come up in the event that the Boolean statement ends up being false.
- Nested if statement: you can utilize one if or if else statement within another one when required.

A few instances of how this functions incorporates:

age = 23

if (age = = 23):

print("The age is 23")

print("Have a good day!")

At the point when this code comes through it will state:

The age is 23

Have a good day!

Utilizing the "if" keyword is going to tell the compiler that what you are composing is a Decision control instruction. The condition that is behind the keyword if it is within the parenthesis, the conditions should be true in the event that you need the code to come out, however on the off chance that it isn't correct, this statement is simply going to be overlooked in this circumstance and it is going to proceed onward to the next command that you give. You can as well set up an "if… else" clause. This one would show the unique message if the conditions were true. In any case, if the conditions are false, it would end up putting out a second statement. This can assist with guaranteeing that you are getting the correct message across regardless of what another person is sending over and keep the interpreter from totally overlooking this step.

So the following inquiry that you might be asking is the means by which we can tell whether the statement is true or false? You should utilize a portion of the relational operators to make this happen on the grounds that it will have the option to compare the two values to check whether they are equivalent, inconsistent, or another choice. A portion of the options that you can use to check whether a statement is correct include:

Expression	Condition for the expression to be true		
X == y	X is equal to y		
X != y	X is not equal to y		
X < y	X is less than y		
X > y	X is greater than y		
X <= y	X is less than or equal to y		
X >= y	X is greater than or equal to y		

Below is a good example

```
age = int(input("Enter your age:"))

if (age <=18):
print("You are not eligible for voting, try next election!")
print("Program closes")
```

This code will print you out a difference example based of the age that is placed in. Since it is an if statement in particular, you will find a solution exactly when the number is 18 or under. Here we will take a look at the output based on the age of 18 and that of 35.

Enter your age: 18

You are not qualified for voting, try next election!

Program ends

Enter your age: 35

Program closes

Presently we should investigate including various statements in your "if" statement. It isn't extraordinary to discover at least two statement being put inside an expression and working fine and dandy if everything is fulfilled. On the off chance that these statements are executed, you will need to ensure that you indent

them appropriately. We should investigate how this could work. Ensure to check out this on your interpreter to get some knowledge in composing code and how the "if" statements are going to work.

```
bonus = 0.0

currentyear = int(input("Enter current year:"))

yearofjoining = int(input("Enter year of joining:"))

yearofservice = currentyear – yearofjoining

if(yearofservice >2):

reward = 1500

print("Bonus - %d" %bonus)

print("Congratulations! We can give you a reward o %d" %bonus).
```

Now if the "if" statement ends up being higher than two, you will find that the congrats statement is going to be displayed. However, in the event that the sum is lower than two, you are going to wind up without any statement since the entirety of the statements were not met. You can put whatever numbers that you need inside it so as to get it to work for each employee in this option.

The "if-else" Statement

So far we have recently been discussing the "if" statements. These are ones that should be true before any of your statement come out. On the off chance that things come out to be false, there will be no statement by any means. Now assuming you want to have different statements displayed then you will need the "if else" statement, you can pick two or more, statements that are going to come up dependent on the outcomes. On the off chance that your outcomes end up being true, you will have the first statement appear, however on the off chance that the outcomes wind up being false, you can pick another statement that you might want to appear also. This guarantees you are getting an answer regardless of the outcome with the goal that the program shows something new. You simply need to ensure that subsequent to working out your "if" statement, you include the "else" and afterward put in the statement that you might want to have displayed. This can take a touch of time to work, yet it opens up such a significant number of extraordinary options with your code to ensure that everything

levels out and looks pleasant with your code.

The elif Statement

Another alternative that you can do with your statements is the elif statement. This one is going to give you the choice of looking at a couple of expressions as true, instead of just one expression as true, with the goal that you can execute the entire block of code once just one of the conditions goes up to be true. Obviously, doing this is discretionary, yet there is the advantage of being able to have any number of elif statements after the if.

How about we investigate what the whole of this is going to look like when you work it out in the syntax:

```
if expression1:
statement(s)
elif expression2:
statement(s)
elif expression3:
statement(s)
else:
statement(s)
```

You will at that point have the option to put your data into the parts and find the solution that is

listed with every one of the parts. For example:

```
Print("Let's enjoy a Pizza! Alright, let’s go inside Pizzahut!")
print("Waiter, Please select Pizza of your choice from the menu")
pizzachoice = int(input("Please enter your choice of Pizza:"))
if pizzachoice == 1:
print('I want to enjoy a pizza napoletana')
elif pizzachoice == 2:
print('I want to enjoy a pizza rustica')
elif pizzachoice == 3:
```

print('I want to enjoy a pizza capricciosa')

else:

print("Sorry, I don’t need any of the listed pizza's, it would be ideal if you bring a Coca Cola for me.")

you will see the data that is listed after print appear for the initial three lines. After the third line, the program ought to ask you to place in one of the four options. Depending on which option you pick, you will be able to see the correct answer, either the pizza of your choice or the option to simply get a beverage. It might resemble somewhat of a wreck from the start, yet it causes you to get alternatives that the code will have the option to comprehend while restoring the right "pizza type" to you all the while.

The "if" statements will give you a great deal of help when you are looking to make some extraordinary things occur in your code, you can likewise utilize nested if statements that lets you to include a couple of these inside one another, adding some more power to your code. Obviously, it might take some understanding to get this all down, yet you are going to be amazed at all that you can do when you are chipping away at your codes.

Loop Control Statements

Up until now, we have talked about a great deal of programs and what you can do when you are working in Python, yet these are on the whole going to be either choice or sequencial control instruction. For the initial ones, we were doing calculations that will be done in afixed order while with the subsequent one, the correct set of instruction were executed based on the result of the conditions that were tried. There were a few confinements as a result of the manner in which they are executed and they are as it were able to play out precisely the same series of action, consistently similarly, and they are just ready to do it one time.

There are times when you will need to work out a code that can be more entangled. One of these options is for the loop statement. This sort of statement will permit the programmer to execute a statement, or even a batch of statement, a few times. On the off chance that you have a statement that you might want to keep returning in the program, you will need to make your own loop statement to get this going. It is conceivable to utilize Python so as to hand these loop statements and there are three strategies that you can pick so as to cause the loop statement to occur. These three techniques include:

- Nesting loops
- Using a for loop
- Using a while loop

The While Loop

The first sort of loop that we will take a look at is the while loop. This is a decent one when you want the code to accomplish something for a fixed number of times. You would prefer not to have it go on uncertainly, however you would like to have it go for a certain amount of times, for example, eight times, before stopping. Study the example below for more knowledge on how the while loop works.

```
#calculation of simple interest. Request that user input principal, rate, number of years.

counter = 1

while(counter < = 3):
```

```
principal = int(input("Enter the principal amount:"))
numberofyeras = int(input("Enter the number of years:"))
rateofinterest = float(input("Enter the rate:"))
simpleinterest = principal * numberofyears * rate/100
print("Simple interest = %.2f" %simpleinterest)
#increase the counter by 1
counter = counter + 1
print("You have calculated simple interest for 3 time!")
```

The result is going to come out to permit the user to put in the data that they want to compute. It will make sense of the interest rate and the amount that will be determined dependent on the numbers that you give. It will go on a loop with the goal that they are able to do this multiple times for utilization of this. You can set it up to take on more if you would like, however for this one we are simply utilizing it 3 times for straightforwardness.

The "For" Loop

This is for loops that you will utilize when you need a piece of code to rehash a certain amount of times. It is somewhat not quite the same as the one above in light of the fact that it is a greater amount of the conventional approach to get things done, however it can in any case be valuable. This alternative will be a piece not the same as what you will discover in C++ and C.

Instead of this loop allowing the user to characterize the stopping condition or the iteration step, Python will have the statement iterate over the items in the order that they appear in the statement. Below is an instance:

```
# Measure some strings:
words = ['apple', 'mango', 'banana', 'orange']
for w in words:
print(w, len(w))
```

At the point when it experiences the loop with this, you will get the four fruit words above turn out in the order that they are composed. On the off chance that you need them to be done in an alternate

order, you should submit them in an alternate order when you are putting them into the syntax, to keep things simple. You won't have the option to make changes to the words when they are in the syntax like above.

Assuming you need to iterate a little more than a particular sequence of numbers, utilizing the Function range() can truly prove to be useful with this. It will create an entire list containing some of the number arithmetic progressions that you are hoping to utilize.

Nested loop

This is fundamentally just one loop that is within another and it will keep running until both of the programs are finished. This can be helpful for various things that you need to do with your program, however below is an example that will give out the multiplication table going from 1 to 10:

#write a multiplication table from 1 to 10

For x in xrange(1, 11):

For y in xrange(1, 11):

Print '%d = %d' % (x, y, x*x)

when you get the result of this program, it will appear to be like this:

1*1 = 1

1*2 = 2

1*3 = 3

1*4 = 4

As far as possible up to 1*10 = 2

At that point it would proceed onward to do the table by twos, for example, this:

2*1 =2

2*2 = 4

Etc until you end up with 10*10 = 100 as your last spot in the sequence. These loops can assist you with getting various things to appear on your system, sometimes uncertainly, however they are

going to prop up through the loop just for the amount of times that you might want it to. You will have the option to put these loops in with a touch of training and there are such a large number of things that you can do. For instance, the simple formula above will give you the whole multiplication table beginning with 1*1 and finishing with 10*10. For such a straightforward statement, you are getting an incredible measure of data from it and a significant number of the loops that you work with will be the same.

Functions

Functions are another significant part of learning the Python language. These are fundamentally blocks of code that are independent and will have the option to play out some sort of rational task in your code. At the point when you set aside the effort to characterize a function, you will be able to specify the name of the function just as the sequence of the statement. You will at that point have the option to call up the function utilizing its name. There are two kinds of functions that we are going to use here and we will talk about them beneath.

User characterized capacities

With this one, you will have the option to characterize the function that you are utilizing. They are going to have essentially similar guidelines that you found with variable names. This implies using underscore characters, numbers, and letters will work extraordinary, however you ought to never utilize a number for the first character in the function. You can't utilize your keywords as the name of your function and you ought to be cautious about having a function and a variable that have a similar name. The empty brackets that you will see after the name will show that the function isn't going to take on any arguments. The first line will be known as the header while the rest are known as the body. You have to ensure that the header closes with a colon and that you indent in the body so the interpreter recognizes what you are doing. A good instance of the syntax or a function is:

def functionname(arg1, arg2, arg3):

'"docstring of the function i.e., a short presentation about the function"

#code inside the function

Return[value]

#a function might not have any arguments or parameters like

def functionname():

#code inside the function

Utilizing the parameters of a function can be substantial in any of the data types that you are utilizing with Python, regardless of whether you are managing user characterized classes, dictionaries, tuple, list, float, and int. We should investigate a portion of the various pieces of this function so you can comprehend the significance of every one and how they work.

Docstring function

. The first string that you see directly after the header in a function is known as the docstring, something that is short for documentation string. It will be utilized in the code so as to clarify what the function does. This is a discretionary part, however it is a decent practice to get into. In the event that you are anticipating having this go over a couple of lines, you ought to consider doing the triple quotes so as to let the system know what you are doing.

The return statement

The function is continually going to give back a value. The return statement will be utilized as a exist function and will return to where it was called from. This statement is permitted to contain expressions that it can evaluate before giving out a value. If there are no expression in the statement, or the return statement isn't even present within the function, you will find that the return you get is the None object.

Lifetime and scope variables

Something else that you can find out about the Python language is about lifetime and scope variable. The scope variable is the part that will be seen and obvious. The variables and the parameters that are in the capacity won't be obvious from the outside eye (the individuals who are taking a look at the code when it is being executed) yet they will have a local scope that will be displayed.

On the other hand, a lifetime is the timeframe that the variable is going to exist in the memory. The lifetime of most variables within functions will be as long as your function executes. After the function returns the value, these will be decimated with the goal that you can place in no information sources and get various outcomes. For instance, in the event that you input certain numbers and, your return winds up being five, next time you're able to put a different information and you may get a six. These will erase after each function has gone through with the goal that you can get new outcomes if new data is entered.

Pass by references

In Python, the entirety of the parameter that you put in will be passed by reference. This implies that the location of the area in the memory will be referenced to the memory area. So in the event that you are going to change what your parameter is able to refer to in the function, this equivalent change is then going to reflect back when you are doing the calling function.

Flow of execution

The execution of the statement is continually going to begin directly at the first statement that is in the program. Your statement will be done just one at a time to stay away from disarray and ensure that the program is running as easily as it should. The execution is likewise going to happen going from top-down so you have to ensure that you are placing them rightly. Ensure that you are characterizing the function before you choose to ring them to get the correct request.

Anonymous functions

Python permits the developer to make anonymous functions. These are functions that won't be bound to a name at run time and you are going to need to utilize a construct that is known as "lambda." This operator is an approach to create these functions that try not to have a name. Fundamentally you will need to make these when the functions are considered expendable, or they are simply required right where they have been made. There are a couple of functions that Lambda functions will work with including reduce(), map(), and filter().

The syntax that you will need to use with the lambda function is:

lambda argument_list: expression.

The reduce, filter, and map function

We have discussed various functions so far in this book, yet it an opportunity to take a look at a Few others that are going to help you with get more out of the code that you are composing. Especially, we are going to study the map, filter, and reduce functions, with list comprehension to assist you getting started.

Map function

The map() function is the one that will apply to every part in an iterable, or those within a list. Generally, you would utilize the anonymous inline function to make this work, however this is one of them that you can use within any of your functions. So if you are attempting to take a shot at a list, this would be a decent one for you to utilize in your system code.

Filter function

Following up is the filter function. Much the same as you would figure from the name, the filter can remove the components in the sequence for which the functions returns a true. The rest would be overlooked. This implies you will need to do a grouping that would be either true or false so as to get this one to function. At the point when the sequence has a couple of alternatives that are valid, the filter function will choose those ones. Though, if every one of them, or possibly a portion of the sequence points, are false, you will find that these are not separated out.

The reduce function

This one is somewhat special. It will take a few values to the sequence and will work it until you end up with only one value, as opposed to the enormous list that you have. It is going to work from left to right so as to find the solutions that you are searching for. Depending on the length of your sequence, you may need to accomplish some work with this to get everything done.

We should take a look at the numbers 1, 2, 3, and 4. The reduce function is going to take these four numbers and transform them into one. It will do this by gathering the 1 and the 2 into a single unit to get 3, at that point gathering the 3 and the 3 into a single unit to get six, and afterward gathering the 6 and the 4 into a single unit to get 10. So the answer will end in a single value of 10. A decent syntax to show how this works is

```
from functools import lessen

results = reduce( (lambda x, y: x +y), [1, 2, 3, 4])

print(result)
```

List comprehension

This is an extraordinary method to make a few lists within Python. A portion of the normal applications will utilize your components to make the new list and others will make a subsequence of these components when they satisfy certain conditions. The list comprehension feature can really be a substitute to utilizing the lambda function also as the other functions that we just discussed. This is on the grounds that the list function is simpler to work with and comprehend. The syntax of utilizing the list comprehension is

```
[expression-involving-loop-variable for loop-variable in sequence]
```

This is going to step right over all the components of the sequence so you can set up the loop variable for every component each in turn and afterward it totally disposes of what you required the lambda function for in the first place.

Conclusion

Working in Python can be a standout amongst other programming languages for you to pick. It is easy to use for even the computer illiterate, yet it has the right force behind it to make it an incredible programming language regardless your programming level or skill. There are simply such a large number of things that you can do with the Python program, and since you can blend it in with a portion of the other programming languages, there is nearly nothing that you can't do with Python on your side. It's anything but an issue in the event that you are truly constrained on what you can do when utilizing a programming language. Python is an incredible path for you to use so as to acclimate and to do some truly astounding things without getting terrified at how all the code will look. For certain individuals, a large portion of the dread of utilizing a programming language is the way that it is difficult to study and understand all the brackets and other complex issues. In any case, this isn't an issue with regards to utilizing Python in light of the fact that the language has been wrapped up to help everybody read and take a look at it together.

This manual will have all the tools that you have to hit the further developed parts of Python. Regardless of whether you are seeing this book since you have a touch of experience utilizing Python and you need to do a couple of things that are more advanced, or you are beginning as a novice, you make certain to discover the appropriate responses that you need in the blink of an eye. So read this manual and discover everything that you have to know to get some incredible codes while utilizing the Python programming.

Machine learning

What is machine learning?

Machine learning is the act of programming systems to study and learn from data. For instance a system with artificial intelligence will be able to determine when an information is true/ false, important or spam, etc.

Why machine learning?

Assuming you prefer to write a filter program without machine learning procedures, you will have to study the problems then write rules and evaluate them to ensure they are as expected after which you launch the program if there where error you then have to analyze the problem and repeat the rest processes and launch again, you observe that every step is done manually unlike when using artificial intelligent. Also machine learning procedures are used to solve more complex problems automatically, lastly machine learning will assist us to learn why machine learning algorithms makes it easy for us to see what we have learnt.

When would it be a good idea for you to utilize machine learning?

• When you have a problem that requires many not insignificant list of rules to discover the solution. For this situation, AI procedures

can disentangle your code and improve execution.

• Very perplexing issues for which there is no solution with a customary approach.

• Non-stable environments: Machine learning programming can adjust to new information.

Sorts of Systems of Machine Learning

There are various types of machine learning systems. We can partition them into classes, based upon whether

• They have been trained with humans or not

- ○ Supervised
- ○ Unsupervised
- ○ Semi-administered
- ○ Reinforcement Learning

• If they can adapt gradually

• If they work just by contrasting new information to discover data points, or can identify new patterns in the information,and afterward will manufacture a model.

What is Supervised Machine Learning?

Most practical machine learning uses supervised learning this type of machine learning, you train the machine using data which is well "**labeled** ." Meaning that some data are already tagged with the correct answer, supervised machine learning is similar to learning which takes place in the presence of a supervisor.

A supervised learning algorithm learns from labeled training data (where you have input variables [X] and an output variable [Y]

{Y = f(X)}, which helps you to predict outcomes for unforeseen data. Successfully building, scaling, and deploying accurate supervised machine learning Data science model takes time and technical expertise from a team of highly skilled data scientists. Although, Data scientist will rebuild models to make sure the insights given remains true until its data changes.

Why Supervised Learning?

Below, are basic reasons for using supervised Learning

- Supervised learning helps you to solve various types of real-world computation problems.
- Supervised learning permits you to collect data or produce a data output from the previous experience.
- Assists you to optimize performance criteria using experience

How Supervised Learning works

Assuming, you want to train a machine to help you predict how long it will take you to drive home from your workplace. Here, you start by creating a set of labeled data. This data includes

- Time of the day
- Weather conditions
- Holidays

All these details are your inputs. The output is the time it took you to drive back home on that day.

As a human you know that if it's raining outside, then the amount of time spent to drive home will increase. But the machine depends on data and statistics to co relate outcomes. Let's see how you can develop a supervised machine learning model of this example which helps the user to determine the commute time. The first thing you need to create is a training data set. This training set will contain the total commute time and corresponding factors like weather, time, holidays etc. Based on this training set, your machine might see there's a direct relationship between the amount of rain and time you will take to get home. So, it ascertains that the more it rains, the longer you will be driving to get back to your house. It might also see the connection between the time you leave work and the time you'll be on the road. The closer you're to 6 p.m. the longer time it takes for you to get home. Your machine may find some of the relationships with your labeled data. This is the beginning of your Data Model, it starts to reason how rain impacts the way people drive. It also starts to understand that more people travel during a particular time of day.

Types of Supervised Machine learning method

Regression: Regression method predicts an output value using training data. For instance: You can use regression to predict the house price from training data. The input variables will be locality, size of a house, etc. then a regression problem has an output variable that has a real value.

Classification:

Classification means to group the output inside a class. If the algorithm attempts to label input into two distinct classes, it is called binary classification. Selecting between more than two classes is referred to as multiclass classification. For instance: Determining whether or not someone will be a defaulter of the loan.

Supervised machine learning algorithms

Some popular examples of supervised machine learning algorithms are:

- Linear regression for regression problems.
- Random forest for classification and regression problems.
- Support vector machines for classification problems.

Benefit of this model : Outputs always have a probabilistic interpretation, and the algorithm can be regularized to avoid overfitting.

Disadvantage of this model : Logistic regression may underperform when there are multiple or non-linear decision boundaries. This method is not flexible, so it does not capture more complex relationships.

Unsupervised Learning

Unsupervised learning is a machine learning technique, where there no need to supervise the model. Instead, you will allow the model to work on independently to predict accurately (here there is absence of output variable). Unsupervised leaning majorly deals on unlabelled data, unsupervised learning algorithms permits you to perform more complex processing tasks than supervised learning. Also unsupervised learning can be more unpredictable compared with other natural learning deep learning and reinforcement learning methods.

Why Unsupervised Learning?

Here, are basic reasons for using Unsupervised Learning:

- Unsupervised learning finds all kind of unknown patterns in data.

- It is simple to get unlabeled data from a computer than labeled data, which needs manual intervention.
- Unsupervised methods help you to find features that can be useful for categorization.
- It occurs in real time, so all the input data to be analyzed and labeled in the presence of learners.

How Unsupervised Learning works?

Using the case of a baby and her family pet (dog) as an illustration, the baby knows and identifies this dog. A few weeks later a family friend brings along a dog and tries to play with the baby. Though the baby has not seen this dog earlier, but he recognized many features (2 ears, eyes, walking on 4 legs) as part of her pet dog. She identifies a new animal like a dog. This is naturally unsupervised learning, where you are not taught but you learn from the data (in this case data about a dog.) Assuming what happened was supervised learning; the family friend would have told the baby that it's a dog.

Types of Unsupervised Machine Learning Techniques

Unsupervised learning problems further grouped into clustering and

association problems. **Clustering**

Clustering is a basic concept when it comes to unsupervised learning. It mainly deals with finding a structure or pattern in a collection of uncategorized data. Clustering algorithms will process your data and find natural clusters(groups) if they exist in the data. You can also change how many clusters your algorithms should identify. It allows you to adjust the granularity of these groups.

Association

Association rules permits you to establish associations amongst data objects inside large databases. This unsupervised technique is about discovering exciting relationships between variables in large databases. For example, people that buy a new home most likely to buy new furniture.

Other Examples:

- A subgroup of cancer patients grouped by their gene expression measurements
- Groups of shopper based on their browsing and purchasing histories
- Movie group by the rating given by movies viewers

Unsupervised learning algorithms

Some popular examples of unsupervised learning algorithms are:

- Apriori algorithm for association rule learning problems. The Apriori algorithm is used in a transactional database to get frequent item sets and then generate association rules. It is commonly used in market basket analysis, where one checks for combinations of products that frequently co-occur in the database. In general, we write the association rule for 'if a person purchases item X, then he purchases item Y' as : X -> Y. Example: if a person purchases milk and sugar, then she is likely to purchase chocolate powder. This could be written in the form of an association rule as: {milk,sugar} -> chocolate powder. Association rules are generated after crossing the threshold for support and confidence.

- k-means for clustering problems. K-means is an iterative algorithm that groups similar data into clusters. It calculates the centroids of k clusters and assigns a data point to that cluster having least distance between its centroid and the data point.
- Principal Component Analysis (PCA) is used to produce data easy to explore and visualize by reducing the number of variables. This is done by capturing the maximum variance in the data into a new coordinate system with axes called 'principal components'.Each component is a linear combination of the original variables and is orthogonal to one another. Orthogonality between components indicates that the correlation between these components is zero.The first principal component captures the direction of the maximum variability in the data. The second principal component captures the remaining variance in the data but has variables uncorrelated with the first component. Similarly, all successive principal components (PC3, PC4 and so on) capture the remaining variance while being uncorrelated with the previous component.

Semi-Supervised Machine Learning
When you have a large amount of input data (X) and only some of the data is labeled (Y) you are doing with semi-supervised learning problems. These problems are in between both supervised and unsupervised learning and this is where common machine language problems fall into. An example of semi supervised machine learning is a photo archive where few of the images are labeled, (e.g. house, plant, animal) and the majority are unlabeled.

Reinforcement learning :

Reinforcement learning is a sort of machine learning algorithm that

allows an agent to decide the best next action based on its current state by learning behaviors that will maximize a reward.

Reinforcement algorithms usually learn optimal actions through trial and error. Imagine, for example, a video game in which the player needs to move to certain places at certain times to earn points. A reinforcement algorithm playing that game would start by moving randomly but, over time through trial and error, it would learn where and when it needed to move the in-game character to maximize its point total.

Batch Learning

In this sort of machine learning, the system can't learn gradually: the system must get all the required information . That implies it will require numerous resources and an enormous measure of time, so it's constantly done offline. Along these lines, to work with this sort of learning, the principal activity is to train the system, and afterward dispatch it with no learning.

Online Learning

This sort of learning is something contrary to batch learning. I imply that, here, the framework can learn gradually by furnishing the system with all the accessible information as instances (individually or groups), and afterward the framework can learn at once. You can utilize this sort of system for issues that require the constant progression of data, which needs to adjust rapidly to any changes. Additionally, you can utilize this sort of system to work with enormous data indexes, You should know how quick your system can adjust to any adjustments in the information's "learning rate." If the speed is high, implies that the system will learn very, rapidly, however it likewise will overlook old information rapidly.

Instance based learning

This is the least difficult kind of discovering that you ought to learn by heart. By utilizing this kind of learning in any program, it will execute any command given to it.

Model-based learning

This kind of learning is based on examples where the system makes prediction from the available examples.

Insignificant Features

The system might have the option to learn if the training information contains enough features and information that aren't excessively insignificant. The main piece of any machine learning

project is to grow acceptable features "of feature engineering".

Feature Engineering

The cycle of feature engineering goes this way:

1. Selection of features: choosing the most valuable features.
2. Extraction of features: consolidating existing features to give more valuable features.
3. Production of new highlights: making of new highlights, in light of information.

System Testing

In the event that you'd prefer to ensure that your model is functioning admirably and that model can sum up with new cases, you can evaluate new cases with it by putting the model in the environment and afterward observing how it will perform. This is a great technique, yet on the off chance that your model is lacking, the user will grumble. You should isolate your data into two sets, one set for training and the second one for testing, so you can train your model utilizing the first and test it utilizing the second. The speculation error is the rate of blunder by evaluation of your model on the test set. The value you get will let you know whether your model is acceptable enough, and in the event that it will work appropriately. In the event that the error rate is low, the model is acceptable and will perform appropriately. Conversely, in the event that your rate is high, this implies your model will perform poorly and not work appropriately. My recommendation to you is to utilize 80% of the data for training and 20% for testing purposes, so that it's extremely easy to test or assess a model.

Overfitting the Data

Assuming you visited a home and one of them steals from you, why leaving the house you will have the thought that every member of that family is a thief. This is an overgeneralization, and, in machine learning, is designated "overfitting". This implies that machines do something very similar: they can perform well when they're working with the training data, yet they can't sum them up appropriately. For instance, in the accompanying figure you'll locate a high level of life fulfillment model that overfits the data, yet it functions admirably with the training data.

When does this happen?

Overfitting happens when the model is extremely complex for the amount of training data given.

Solutions

To take care of the overfitting issue, you ought to do the accompanying:

- Gather more data for "training data"

- Reduce the commotion level

- Select one with fewer parameters

Underfitting the Data

From its name, underfitting is something contrary to overfitting, and you'll experience this when the model is exceptionally easy to learn. For instance, utilizing the case of personal satisfaction, real life is more complex than your model, so the expectations won't yield the same, even in the training models.

Solution

To fix this issue:

- Select the most powerful model, which has numerous parameters.

- Feed the best highlights into your algorithm. Here, I'm alluding to feature enginerring.

- Reduce the constraints on your model.

Chapter 2

System classification
For you to appreciate this chapter you must install python, matplotlib and scikit-learn

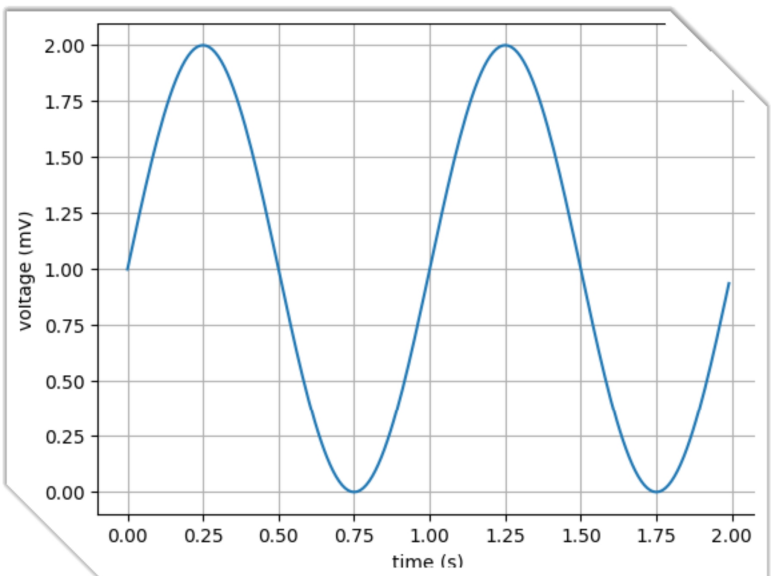
Matplotlib python

Matplotlib is a plotting library for the programming language; it provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits.

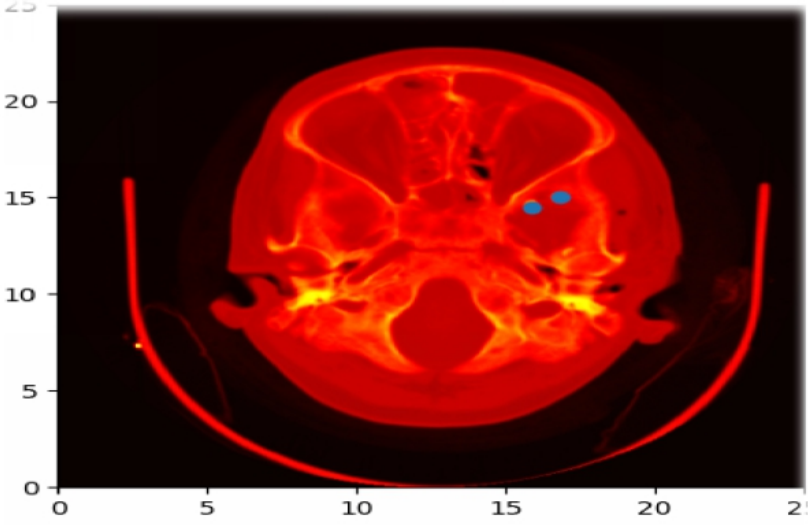
Examples of plots in metplotlib

Below are few samples of plots in metplotlib.

Line Plot



Images
Matplotlib can display images



Arbitrary path
Metplotlib has option for arbitrary paths

Scikit-learn is a machine learning library for Python. It features different algorithms like support random forests, and k-neighbours, vector machine, and it also supports Python numerical and scientific libraries like numpy and scipy.

The Mnist

In this part, you'll go further into grouping systems, and work with

the Mnist data set. This is a group of 70,000 pictures of digits handwritten by students and employees. You'll see that each picture has a label and a digit that represents it. This task resembles the "hello, world" case of customary programming. So every learner to AI should begin with this venture to learn about the grouping calculation. Scikit-Learn has numerous capacities, including the MNIST. How about we take a look at the code:

>>> from sklearn.data sets import fetch_mldata

>>> mn= fetch_mldata('MNIST original)

>>> mn

{'COL_NAMES': ['label', 'data'],

Description: 'mldata.org data set: mn-original,

data: array([[0, 0, 0,..., 0, 0, 0],

[0, 0, 0,..., 0, 0, 0],

[0, 0, 0,..., 0, 0, 0],

...,

[0, 0, 0,..., 0, 0, 0],

[0, 0, 0,..., 0, 0, 0],

[0, 0, 0,..., 0, 0, 0]], dataType=uint8),

'tar': array([0., 0., 0.,..., 9., 9., 9.])} de

. Description is a key that defines the data set.

. The data key here contains an array with only one row for example, and a

column for each feature.

. This target key contains an array with labels.

Let practice with few of the code:

>>> X, y = mn["data"], mn["tar"]

>>> X.shape

(70000, 784)

>>> y.shape

(70000,)

. 7000 here implies that there are 70,000 images, and each picture

has more than

700 features: "784". because, as should be obvious, each picture is 28 x 28 pixels, you can envision that each pixel is one feature.

We should take another instance from the data set. You'll just need to get an

instance’s feature, at that point make it 26 x 26 array, and afterward show them utilizing the imshow function:

```
%matplotlib inline

import matplotlib

import matplotlib.pyplot as plt

yourDigit = X[36000]

Your_image = your_image.reshape(26, 26)

plt.imshow(Your_image, cmap = matplotlib.cm.binary,

interpolation="nearest")

plt.axis("off")

plt.show()
```

As should be obvious in the picture below, it’s like the number five, and we can give that a label that reveals to us it's five.

In the following figure, you can see more mind boggling grouping task from the MNIST data set.

Additionally, you ought to make a test set and make it before your data is assessed.

As earlier mentioned the MNIST data set is in two sets, one for training and the other for testing.

x_tr, x_tes, y_tr, y_te = x [:60000], x[60000:], y[:60000], y[60000:]

How about we play with your training set as follows to make the cross-validation to be comparative (with no missing of any digit)

Import numpy as np

myData = np.radom.permutaion(50000)

x_tr, y_tr = x_tr[myData], y_tr[myData]

Presently it's an ideal opportunity to make it sufficiently basic, we'll attempt to simply recognize one digit, for example the number 6. This "6-detector" will be a case of the binary classifier, to recognize 6 and not 6, so we'll make the vectors for this task:

Y_tr_6 = (y_tr = = 6)// this implies it will be true for 6s, and false for some other number

Y_tes_6 = (Y_tes = = 6)

From that point forward, we can pick a classifier and train it. Start with the SGD (Stochastic Gradient Descent) classifier. The Scikit-Learn class has the capacity of taking care of extremely huge data set. In this example, the SGD will manage occurrences independently, as follows.

from sklearn.linear_model import SGDClassifier

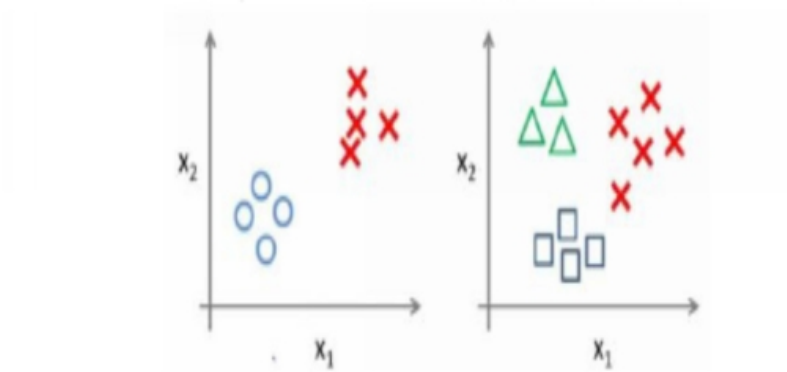
mycl = SGDClassifier (random_state = 42)

mycl.fit(x_tr, y_tr_6)

to utilize it to distinguish the 6

> > >mycl.prdict([any_digit)]

Binary classification multi-class



Performance measures
If you want to evaluate the classifier, this will be more difficult than a regressor, so

Let us explain how to evaluate a classifier.

In this example, we will use cross-validation to evaluate our model.

sklearn.model_selection Import StratifiedKFold

Form sklearn.base import clone

SF = StratifiedKFold (n = 2, run_state = 40)

For train_index, test_index in sf.split (x_tr, y_tr_6):

Cl = no = clone (sagd_claf)

x_tr_fd = x_tr [train_index]

y_tr_fd = (y_tr_6 [train_index])

x_tes_fd = x_tr [test_index]

y_tes_fd = (y_tr_6 [test_index])

Cl.fit (x_tr_fd, y_tr_fd)

y_p = cl.predict (x_tes_fd)

Print (n_correct / len (y_p))

. We use the stratified fold class to do the stratified sampling that generates folds containing rations for each class. Next, ever iteration in the code will form a Clone of the classifier to make predictions on the test fold. And finally, it will calculate the number of correct predictions and their ratio

. We will now use the cross_val_score function to evaluate the SGDC classifier by K-fold cross validation. The k fold cross validation will split the training set into 3 folds, and then it will predict and evaluate on each fold.

Sklearn.model_selection import from cross_val_score

Cross_VaL_Score (Sgd, Clf, X_ Y_Tr_6, CV = 3, Scoring = "Accuracy")

You will find the accuracy ratio of "correct predictions" on the folds.

Let's classify each classifier on each image in the not-6

From sklearn.base Import from Base Estimator

Class never 6 Classifier (Base Estimator):

Def fit (self, x, y = None):

Pass

Def Forecast (Self, X):

Return NP.Zeroz ((lane (X), 1), dtype = bool)

Let's check the accuracy of this model with the following code:

>>> never_6_cl = never6Classifier ()

>>> cross_val_score (never _cl, x_tr, y_tr_6, cv = 3, scoring = "accuracy")

Output: array ([“num”, “num”, “num”])

As for the output, you get no less than 90%: only 10% of the images are 6s, so we

One can always imagine that an image is not a 6. We will be correct about 90% of the time.

Keep in mind that accuracy is not the best performance criterion for classifiers, if

You are dealing with skewed data sets.

Confusion matrix

The confusion matrix is a better way to evaluate the performance of your classifier, measuring performance with confusion matrix, is easy just by calculating the number of times instances of class X are classified as class Y, for example. To have the number of times of image classifiers of 6s with 2s, you should look at the 6th row and 2nd column of the confusion matrix.

Let's calculate the confusion matrix using the cross_val_predict () function.

From sklearn.model_selection Imported from cross_val_predict

y_tr_pre = cross_val_predict (sgd_cl, x_tr, y_tr_6, cv = 3)

This function, such as the cross_val_score () function, performs the K fold cross_validation, it also returns predictions at every fold. It also gives a clean prediction for each instance in your training set.

We are now ready to obtain the metrix using the following code.

From sKlearn.metrix import confusion_metrics

Confusion_metrics (y_tr_6, y_tr_pred)

You will find an array of 4 values, "numbers".

Each row represents a class in the matrix, and each column represents a column predicted class.

The first row is the negative: containing "non-6 images". You can learn a lot from the matrix.

But there is also a good one, which is interesting to work with if you want to get The accuracy of positive predictions, which is the precision of classifier

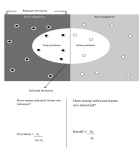
using this equation.

Precision = (TP) / (TP + FP)

TP: The number of true positives

FP: Number of false positives

Recall = (TP) / (TP + FN) "Sensitivity": It measures the ratio of positive instances.



Recall

> > > from sKlearn.matrics import precision _score, recall_score

> > > precision_score (y_tr_6, y_pre)

> > > recall_score (y_tr_6, y_tr_pre)

It is very common to combine precision and recall only one metric, which is the F1 score.

F1 means both precision and recall. We can count the F1 score with this equation:

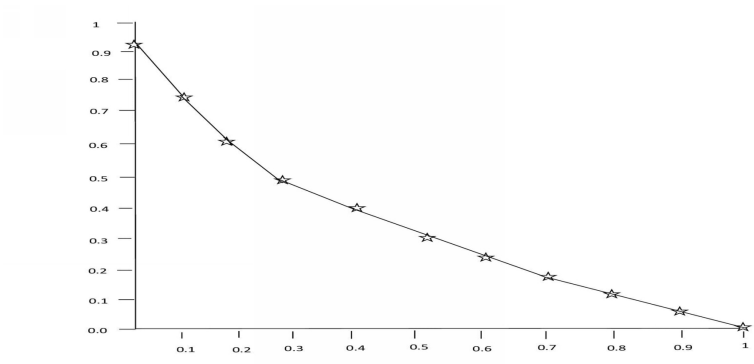
F1 = 2 / ((1 / precision)) ((1) / recall)) = 2 * (precision * recall) / (precision +

Recall) = (TP) / ((TP) + (FN + FP) / 2)

To calculate the F1 score, just use the following function:

> > > from sKlearn.metrics import f1_score

> > > f1_score (y_tr_6, y_pre)

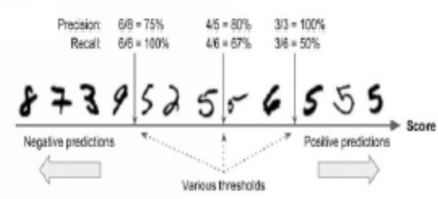


Recall Tradeoff

To get to this point, you should take a look at the SGDC classifier and how it should look(Makes decisions regarding classification). It calculates the score based on that decision function, and then it compares the score with the threshold. if it is more than this score, it will exemplify "positive or negative".class

For instance, if the decision is in the threshold center, you will find 4 true + on the right of the threshold, and only one wrong. So the accuracy ratio will just be

80%.



In SciKit-Learn, you cannot set thresholds directly. You will need access The decision scores, which uses predictions and by y calling the decision function, ().

>>> y_sco = sgd_clf.decision_funciton ([any digit])

>>> y_sco

>>> Threshold = 0

>>> y_any_digit_pre = (y_sco> Threshold)

In this code, the SGDClassifier has a threshold, = 0, to return the same outcome as the predict () function.

> > > Threshold = 20000

> > > y_any_digit_pre = (y_sco > Threshold)

> > > y_ any_digit_pre

This code will prove that, when the threshold increases, the recall decreases.

y_sco = cross_wall_predict (sgd, cl, x_tree, y_tr_6, cv = 3, method = ”decision

Function)

This is the time to calculate all possible precision and remember the threshold by calling the precision_recall_Curve () function

Import from sKleran.metrics the precision_recall_curve

precisions, recall, threshold = precision_recall_curve (y_tr_6, y_sco)

Let's now plot precision and recall using Metplotib

Def Plot_pre_re (pre_re_thr):

plt.plot (thr, pre [: - 1], “b—“, label = “precision”)

plt.plot (thr, re [: 1], "g-", label = "recall")

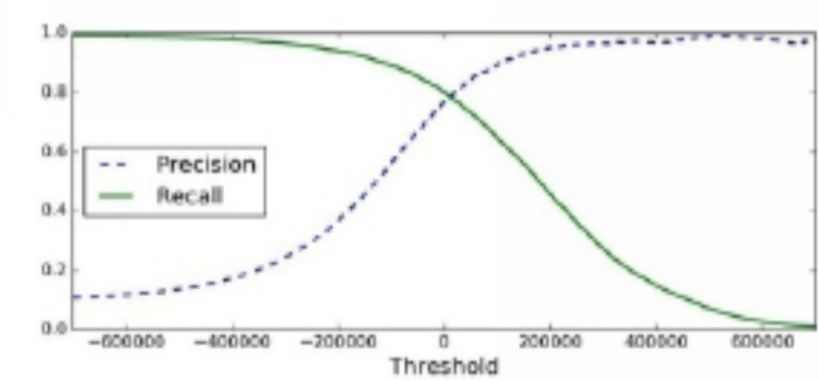
plt.xlabel ("Threshold")

plt.legnd (loc = "left")

plt.ylim ([0,1])

Plot_pre_re (pre, re, thr)

plt.show



ROC

ROC is meant for the receiver operating characteristic and it is a tool that works well with binary Classifieds.

This tool is similar to a recall curve, but does not plot for precision and recall:

It plots the positive rate and negative rates. You will also be able to work with FPR, which is the ratio of negative samples. You can imagine if it is like this (1 - negative rate. There is another concept TNR and that is the specificity. Recall = 1 – specificity).

Let's play with the ROC curve. First, we have to calculate TPR and FPR, just by calling the roc-curve () function,

from sKlearn.metrics roc_curve

fp, tp, thers = roc_curve (y_tr_6, y_sco)

After that, you can plot FPR and TPR with Metplotlib. According to the concept of

The following instructions

Def_roc_plot (fp, tp, label = none):

plt.plot (fp, tp, linewidth = 2, label = label)

plt.plot ([0,1)], [0,1], "k--")

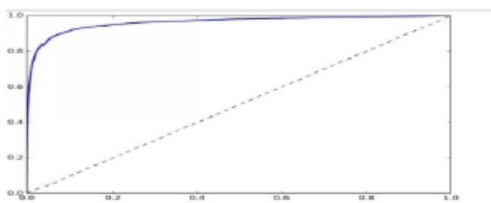
plt.axis ([0,1,0,1])

plt.xlabel (‘This is the false rate’)

plt.ylabel (‘this is the true rate’)

roc_plot (FP, TP)

plt.show



Multi-class classification

We use binary classifiers to differentiate between any two classes, but when you have the task to distinguish between multiple classes you use something like a random forest classifier or a Bayes classifiers, which can compare between more than two. But, on the other hand, SVM (support vector machine) and linear classifiers act like binary classifiers.

If you want to develop a system that categorizes images of digit into 12 classes(0 to 11) You need to train 12 binary classifiers, and create one for eachclassifier (such as 4-detector, 5-detector, 6-detector and more), and then you will need to get DS, "decision score" of each classifier for the image. Then you will choose the highest score classifier. We call this the OvA strategy: one-versus-all.'' The other method is to train the binary classifier for each pair of digits; For instance, one for 5s and 6s and another for 5s and 7s. - We call this method OvO, "one-versus-one" – to count the number of classifiers needed depends on the number of classes that use the following equation: ''N= number of classes'' N * (N-1) / 2. If you want to use this technique with MNIST 10 * (10-1) / 2,The output will be 45 classifiers, "binary classifiers".

In Scikit-Learn, you run OvA automatically when you use binary. classification algorithm.

> > > sgd_cl.fit (x_tr, y_tr)

> > > sgd_cl. predict([any issue])

Additionally, you can call the decision_function () to return scores "10 scores".For a class ”

> > > any_digit_scores = sgd_cl.diction_function ([any_digit])

> > > any_digit_scores

Arrays (["num", "num", "num", "num", "num", "num", "num" "

, ”“ Num ”]])

Training to random forest classifiers

> > > forest.clf.fit (x_tr, y_tr)

> > > Forest.clf.predict ([any digit]]

Array ([num])

As you can see, training with Random Forest Classifier is just two lines of code and its very simple. Scikit-Learn does not run any OvA or OvO functions because this type of algorithm - "Random Forest Classifier" - can perform multiple tasks automatically classes. If you

want to take a look at the list of classifier possibilities, you can call the predict_oroba () function.

>>> forest_cl.predict_proba ([any_digit])

Array ([[0.1, 0, 0, 0.1, 0, 0.8, 0, 0, 0]]))

The classifier is very accurate with its prediction, as you can see in the output; there is 0.8 at the index number 5.

Let's evaluate the classifier using the cross_val_score() function.

>>> cross_val_score(sgd_cl, x_tr, y_tr, cv = 3, scoring = "accuracy")

Array ([0.84463177, 0.859668, 0.8662669])

You will get 84% more times. When using a random classifier, you will find in this case, 10% for the accuracy score. Keep in mind that the higher this value is, the Better.

Error analysis

First of all, when developing a machine learning project:

1. Determine the problem;

2. Collect your data;

3. Work on and explore your data;

4. Clear data

5. Work with many models and choose the best;

6. Connect your models to the solution;

7. Show your solution;

8. Run and test your system.

First, you should work with the confusion metrics and make predictions by cross-val function. Next, you will call the confusion matrix function:

>>> Y_tr_lover = cross_val_prediciton(sgd_cl, x _tr_scaled, y_tr, cv

= 3)

>>> cn_mx = confusion_metrics(y_tr, y_tr_pre)

>>> cn_mx

Array ([5625, 2, 25, 8, 11, 44, 52, 12, 34, 6],

[2, 2415, 41, 22, 8, 45, 10, 10, 9],

[52, 43, 7443, 104, 89, 26, 87, 60, 166, 13]

[47, 46, 141, 5342, 1, 231, 40, 50, 141, 92]

[19, 29, 41, 10, 5366, 9, 56, 37, 86, 189]

[73, 45, 36, 193, 64, 4582, 111, 30, 193, 94],

[29, 34, 44, 2, 42, 85, 5627, 10, 45, 0],

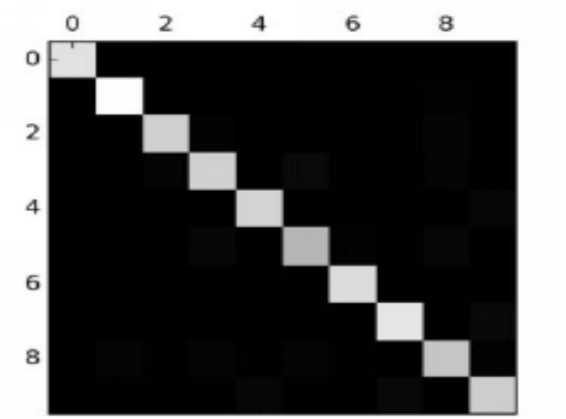
[25, 24, 74, 32, 54, 12, 6, 5787, 15, 236]

[52, 161, 73, 156, 10, 163, 61, 25, 5027, 123]

[50, 24, 32, 81, 170, 38, 5, 433, 80, 4250]]

Plt.matshow (cn_mx, cmap = plt.cm.gray)

Plt.show ()

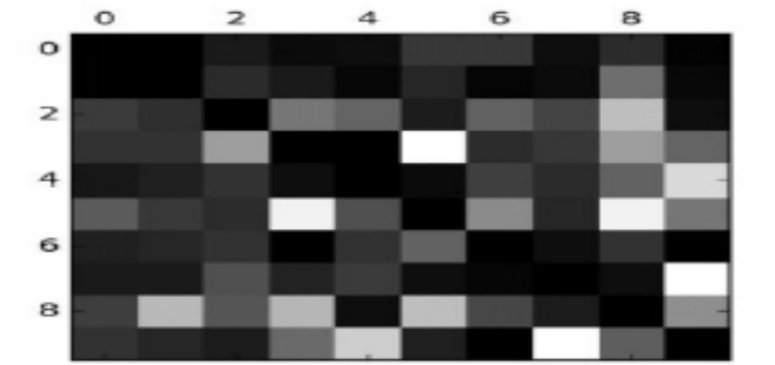


First, you should divide each value in the matrix by the number of images in class, and then you will be able to compare the error rate.

```
rw_sm = cn_mx.sum(axis = 1, keepdims = true)

nm_cn_mx = cn_mx / rw_sum
```

The next step is to make all the zeros on the diagonal, and it forestall the errors from occurring.

```
np.fill_diagonal (nm_cn_mx, 0)

plt.matshow (nm_cn_mx, cmap = plt.cm.gray)

plt.show ()
```



It is easy to see the errors in the schema above. One thing to keep in mind is that the rows represent classes and columns represent the predicted values.

Multi-label classifications

In the above examples, there is only one example in each class, let's say you want to assign the examples to multiple classes (like facial recognition), for example

Suppose you want to find more than one face in a single photo. There will be a label for each face. Let's practice with a simple example.

y_tr_big = (y_tr> = 7)

y_tr_odd = (y_tr% 2 = = 1)

Y_multi = np.c [y_tr_big, y_tr_odd]

kng_cl = KNeighborsClassifier()

Kng_cl.fit (x_tr, y_m, ulti)

In these instructions, we have created a y_multi array that has two labels for each image.And the first contains information on whether the digit is"large" (8,9,) another checks to see if it's not.

Next, we will make predictions using the following instructions.

> > > kng_cl.predict ([any digit])

Array ([false, true], dataType = bool)

True here means it's false, meaning it's not big.

Multi-output classification

At this point, we can cover the final type of classification function, which is multi-output classification.

It's just a common case of multi-label taxonomy, but each label will have one multiclass. In other words, it will have more than one value.

Let's be clarified by this example, using MNIST and adding images some noise to the image with NumPy functions.

No = rnd.randint (0, 101, (lane (x_tr), 785)))

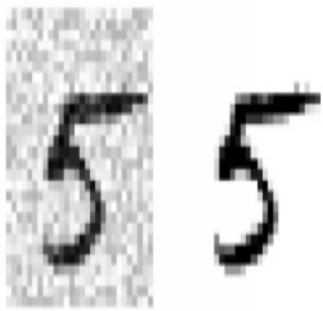
No = rnd.randint (0, 101, (lane (x_tes), 785))

x_tr_mo = x_tr + no

x_tes_mo = x_tes + no

y_tr_mo = x_tr

```
y_tes_mo = x_tes
Kng_cl.fit (x_tr_mo, y_tr_mo)
cl_digit = kng_cl.predict (x_tes_mo [any index]])
Plot_digit (cl_digit)
```



How to train a model

Having gone this far in learning machine learning models and as well training algorithms which at the first sight was quite challenging, out of this fear we managed to penetrate regression system, and also cleared the atmosphere on image classifiers. Although in the previous chapters we achieved more than that but we didn't comprehend the core principles of a model so now we need to progress deeper into the wonder land so that we can understand how models work and how to use them for purposes.

Getting a deeper understanding of the above details will help you find the right model and by choosing the best training algorithm. Also, it will help you with debugging and error analysis. In this chapter, we will deal with polynomial regression, which is a complex model that works for nonlinear data sets. In addition, we will work with some regularization techniques that cuts down training that promote overfitting.

Linear regression

For example, we would take l_S = θ0 + 1 × GDP_per_cap. This is a simple model

For the linear function of the input feature, select "GPD_per_cap". (θ0 and θ1) are the dimensions of the model,

In general, you will use a linear model to calculate and predict

the weighted amount of input features and the constant "bias" as written below

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

. Y is the value of the predictor.

. N represents the features

. X1 is the value of the feature.

. Θj is the model dimension of J Theta.

Also, we can write an equation in vector form like the following example.

$$\hat{y} = h_\theta(\mathbf{x}) = \theta^T \cdot \mathbf{x}$$

. Θ is the value that lowers the price.

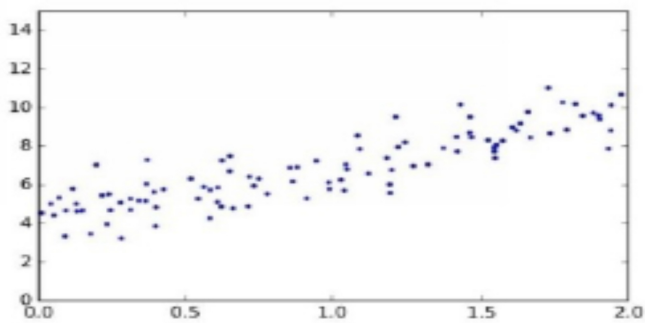
. Y has values from y (1) to y (m).

Let's write some code to practice.

Import numpy as NP

V1_x = 2 * np.random.rand (100, 1)

V2_y = 4 + 3 * V1_x + np.random.randn (100, 1)



After that, we will calculate the value Θ using our equation. It's time to use this inv () function from our linear algebra module of numpy (np.linalg) to calculate the inverse of any matrix, and also, the dot () function for multiplication of our matrix

Value 1 = np.c_ [np.ones ((100, 1)), V1_x]

myTheta = np.linalg.inv (Value1.T.dot (Value 1)). Dot (value 1.T) .dot (V2_y)

> > > my Theta

Array ([[num], [num]])

This function uses the following equations - y = 4 + 3x + sound “Gaussian” -

To generate our data.

Let's make our predictions now.

>>> V1_new = np.array ([[0], [2]])

>>> V1_new_2 = np.c_ [np.ones ((2,1%), v1_new]]

>>> V2_predicit = V1_new_2.dot (myTheta)

>>> V2_predict

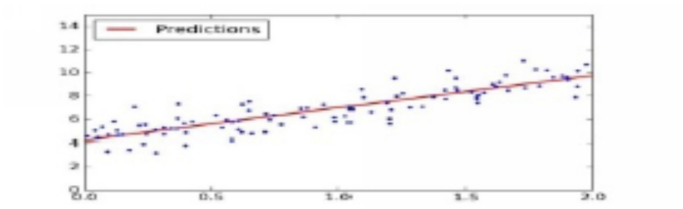
Array ([[4.219424], [9.74422282]])

Now, it's time to plot the model.

Plt.plot (v1_new, v2_predict, "r-")

Plt.plot (V1_x, V2_y, "b.")

Plt.axis ([0,2,0,15])



Plt.show ()

Computational complexity

With the general formula, we can calculate the inverse of MˆT.M - i.e., a

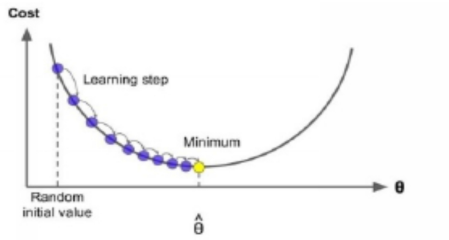
n * n matrix (n = number of features). The complexity of this inversion is something like O (n ˆ 2.5) to O (n ˆ 3.2), depending on the implementation. Indeed, if you create features like twice, you will create the time the calculation comes between 2 ˆ 2.5 and 2 ˆ

The good news here is that this equation is a linear equation. Which means it easily handle a large training set and fit the memory.

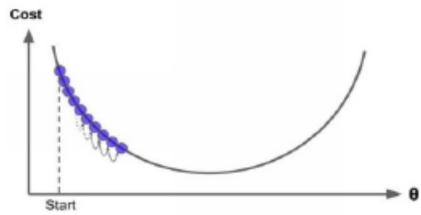
After training your model, predictions will not be slow, and the complexity will be simple, thanks to the linear model. It's time to dig dip and move on into the procedures of training a linear regression model, which is always used when large number of features and instances in the memory.

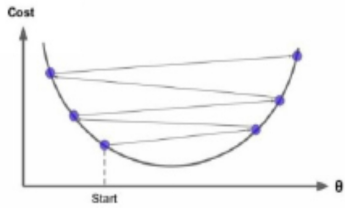
Gradient descent
This algorithm is a common algorithm used for optimization and providing the best solution for various problems. The idea of this algorithm is to work with the parameters in an iterative way, to make the cost function very simple.

The gradient descent algorithm calculates the gradation of error using the parameter theta, and it works with procedure of descending gradient. If the gradient is equal to zero, you will reach the minimum.



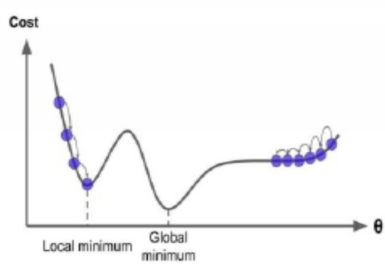
Also, you should bear in mind that the size of the steps is vital for this algorithm, because if small (meaning the rate of learning) is slow, i.e it will take longer time to cover everything that it needs to.



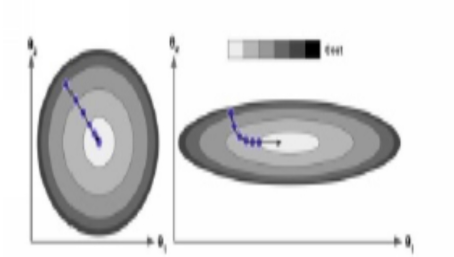


If the rate of learning is high, it will take less time to cover what's needed, and it gives optimal solution.

At times not all cost functions are easy there are some irregular functions that make getting an optimal solution very hard. This problems are detected when the local minimum and global minimum looks like the do in the following figure



If you assign any of the two points on your curve, you will observe that segment of the line will not join them at the same turn. This cost will look like a task Bowl, which will happen if the features have many scales like the image below.



Batch gradient descent
If you want to implement this algorithm, you must first calculate the gradient of your cost function using theta parameters. If the value of the parameter Theta has changed; you will need to know the changing rate of your cost function. We can call this change by a partial derivative

We can calculate the partial derivation using the equations below:

$$\frac{\partial}{\partial \theta_j} \text{MSE}(\theta) = \frac{2}{m} \sum_{i=1}^{m} (\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)})\, x_j^{(i)}$$

But we will also use the following equations to calculate the partial derivatives and the gradient vector together.

$$\nabla_\theta \text{MSE}(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \text{MSE}(\theta) \\ \frac{\partial}{\partial \theta_1} \text{MSE}(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \text{MSE}(\theta) \end{pmatrix} = \frac{2}{m} \mathbf{X}^T \cdot (\mathbf{X} \cdot \theta - \mathbf{y})$$

Let us implement the algorithm.

Lr = 1 # Lr for learning rate

Num it = 1000 # number of iterations

L = 100

myTheta = np.random.randn (2,1)

for it in range (num_it)

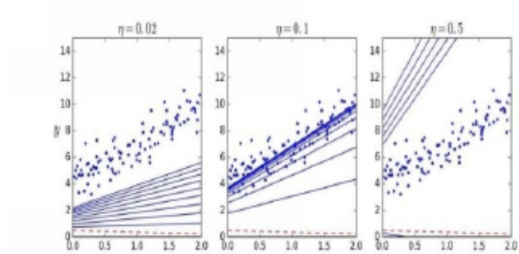
gr = 2 / L * Value1.T.dot (Value1.dot (myTheta) - V2_y)

myTheta = myTheta - Lr * gr

>>> My theta

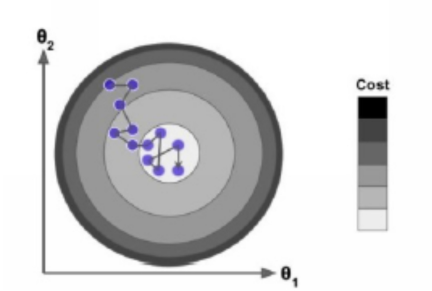
Array ([[num], [num]])

If you try to change the value of the learning rate, you will get different shapes, e.g.



Stochastic gradient descent
You will encounter a problem when you are using a batch gradient descent: it requires the use of the whole training set at each step, and that will affect the performance "speed".

But when using stochastic grad descent, the algorithm will randomly choose an instance from the training set at any step, and then it will count the value. The stated procedure makes the stochastic grad descent faster than the batch gradient descent, knowing that the stochastic gradient descent does not need the whole set to calculate the value. Another thing to take note of is that because of its randomness procedure, it will be irregular when compared to batch algorithm.

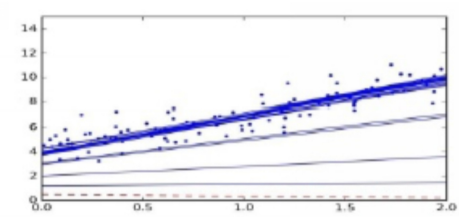


Let us implement the algorithm.

```
Nams = 50
L1, L2 = 5, 50
Def lr_sc (s):
    return L1 / (s + L2)
myTheta = np.random.randn (2,1)
for Num in range (Nums):
For l in range (f)
myIndex = np.random.randint(f)
V1_Xi = value 1 [myindex: myindex + 1]
V2_yi = V2_y [myIndex: myIndex + 1]
gr = 2 * V1_xi.T.dot (V1_xi.dot (myTheta) - V2_yi)
Lr = lr_sc (num * f + i)
myTheta = myTheta - Lr * gr
```

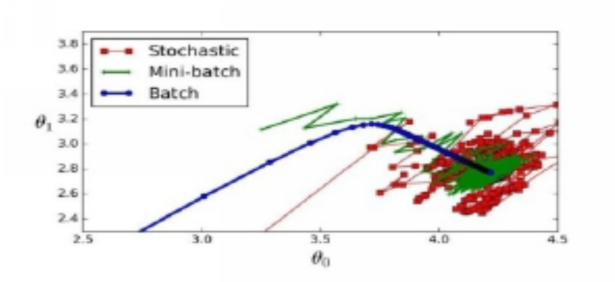
>>> My theta

Array ([[num], [num]])



Mini-batch gradient descent

Because you already know the batch and stochastic algorithms, this kind of algorithms is very easy to understand and work with. As you know, both algorithms, calculate the value of a gradients, based on the whole training set or just one instance. However, the mini-batch calculates its algorithms based on small and random sets, and performs better than the other two algorithms.



Polynomial regression

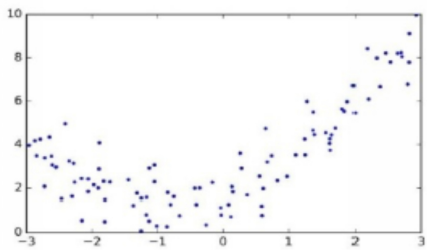
We will use this technique, in particular, when dealing with more complex data. In the case of linear and nonlinear data. After we have add the power of each feature, we can train the model with new features. This is known as polynomial regression.

Now, let's write some code.

L = 100

V1 = 6 * np.Random.Rand (L, 1) - 3

V2 = 0.5 * V1 ** 2 + V1 + 2 + np.random.randon (L, 1)



So from the diagram, the straight can never represent the data in the most efficient way. So we will use the polynomial process to work on this problem.

>>> from sklearn.preprocession Polynomial import polynomial features

>>> PF = multicellular features (degree = 2, inclusion_seed = false)

>>> V1_P = PFFit_Transform (V1)

>>> V1 [0]

Array ([number])

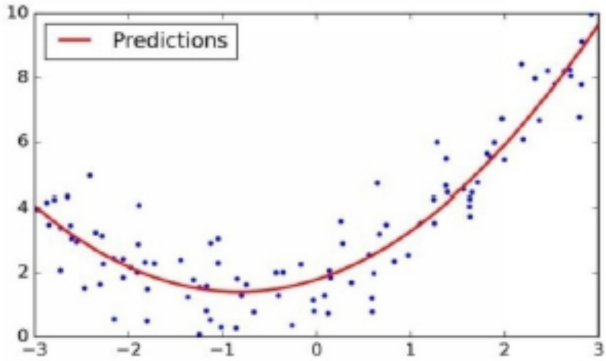
> > > V1_P [0]

Now, let's do this work properly with our data, and change the straight line.

> > > ln_r = Linear Argument ()

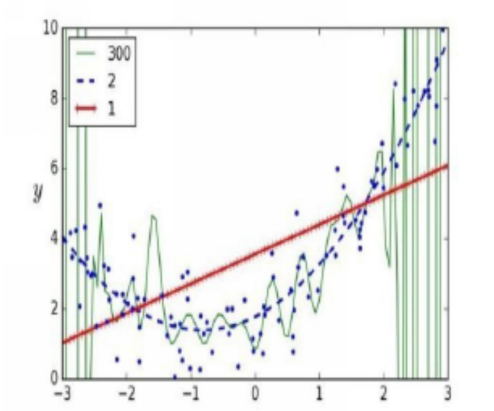
> > > ln_r.fif (V1_P, V2)

> > > ln_r.intercep_, ln_r.coef



Learning curves

Suppose you are dealing with polynomial regression, and you need it to fit the data better than that of linear one. In the image below, you will find 300-degree model. We can also compare the final result with other types of the regression: "normal linear".



In the figure above, you can see the overfitting of data when you're using the polynomial. On the other hand, with linear one, you can see that the data apparently been underfitted.

Regularized Linear models

We have worked in the first and second chapters on how to reduce overfitting by regularizing the model a little, as an example, if you'd

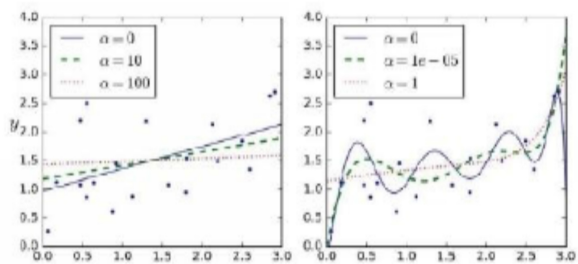
to like to regularize a Polynomial model. In this case, to fix the problem, you should decrease the number of degrees.

Ridge regression

Ridge regression is another version of linear regression, but, then regularizing it and adding weight to the cost function, this makes it fit the data, and makes the weight of the model as smooth as possible. Here is the cost function of the ridge regression:

$$J(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

As an example of ridge regression, just observe the following figures.



Lasso regression

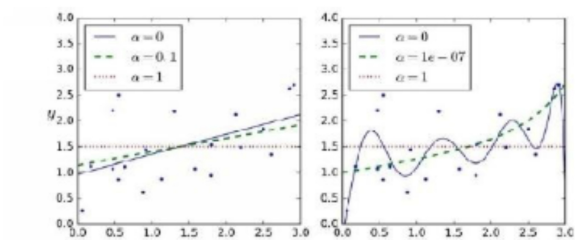
“Lasso” regression stands for “least absolute contraction and selection operator”regression. This is another type of regularized version of the linear regression.

It sounds like a ridge regression, but there is a small difference in the equation, as in the following figures

The cost function of the Lasso Regression Operations:

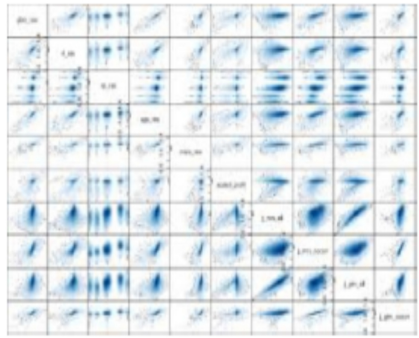
$$J(\theta) = \text{MSE}(\theta) + \alpha \sum_{i=1}^{n} | \theta_i |$$

As you can see in the figure below, Lasso regression uses smaller values than the ridge.



Chapter 4

Combinations of different models



Tree classifiers

The next image illustrates the definition of a common target of collecting functions that are simply for merging different classifiers into one-class including a more generalization performance than each individual classifier alone.

For example, suppose you have collected predictions from many experts. Ensemble methods will allow us to merge these predictions by lots of experts to get a prediction that is stronger than the predictions of each individual expert. As you can see later in this chapter, there are a lot of different methods for creating a combination of classifiers. In this part, we will introduce one basic understanding of how ensemble works and why they are common to achieve a good generalization performance.

In this chapter, we will work with the most popular ensemble method that uses the majority voting principle. A lot of voting simply means we choose the label that has been predicted by most classifiers (more than 50% votes was achieved). For example, the word vote here is just binary class settings only. However, generating a majority voting principle to multi-class setting is not difficult

The principle is multi-class settings, called majority voting. After that, we

The class will select the label that received the most votes. The following figure

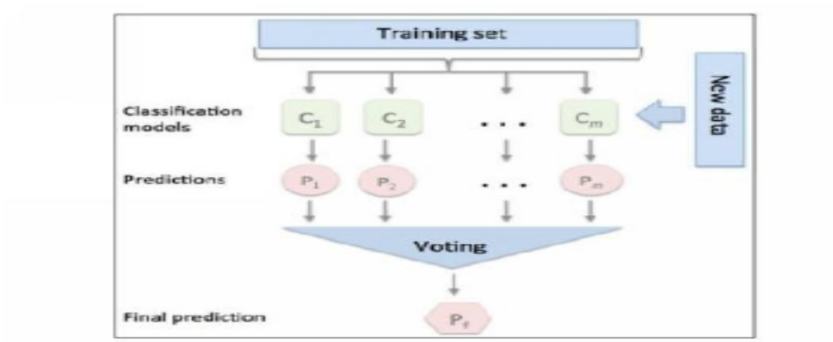
Explains the concept of majority and majority voting for the 10 alliance

Classified where each unique symbol (triangle, square and circle) a

Unique class label:

Using the training set, we start by training m different classifiers (CC1,,…8M).

Depending on the method, the ensemble can be made from several classifications algorithms; For example, decision trees, support vector machines, logistics regression classifier, and so much more. In fact, you can use the same base classification algorithm fitting different subsets of the training set. An instance of this method will be a random forest algorithm, which combines many decisions ensemble ways to use majority voting.



To predict a class label by simple majority or plurality voting, we combine these predicted the class labels of each individual classifier C j and select the class label y^ that got the most votes:

y m ^ = ode {CC 1 2 () x x ,, ()), Cm () x}

For example, in a binary classification function where class 1 1 = - and class 2 1 = +, we can write a majority vote prediction.

Furthermore to explain why ensemble methods will work better than individual classifications alone, let's apply the simple concepts of combinatory. For the following example, we assume that all n base classifiers for binary classification task have the same error rate, ε . Furthermore, we assume that the classifiers are independent and does not relate to the error rate. As you can see, we can only explain the error statistics of an ensembe of the base classifiers as probability.

Mass function of a binomial distribution:
Here, n. k is the binomial coefficient: n choose k. You can see how simple it is to calculate the probability that the prediction of the ensemble is incorrect. Now, let's work out a more solid example of 11 base classifiers (n = 11) with an error rate of 0.25 (ε = 0.25):

You may notice that the error rate of the ensemble (0.034) is lower than the error rate of each individual classifier (0.25) if all assumptions are met. Note that in this simplified picture, a 50-50 split by an even number of classifier n is considered as an error, where this is the correct half of the time. To compare such an idealistic nsemble classifer to a base classifer over a range of dfferent base error rates, let’s implement the probability mass function in python:

> > > Import math

> > > def ensemble_error (n_classifier, error):

... q_start = math.ceil(n_classifier / 2.0)

... probability = [comb (n_class, q) *

... error ** Q *

... (1-error) ** (n_classifier - q)

... for q in series (q_start, l_classifier + 2)]

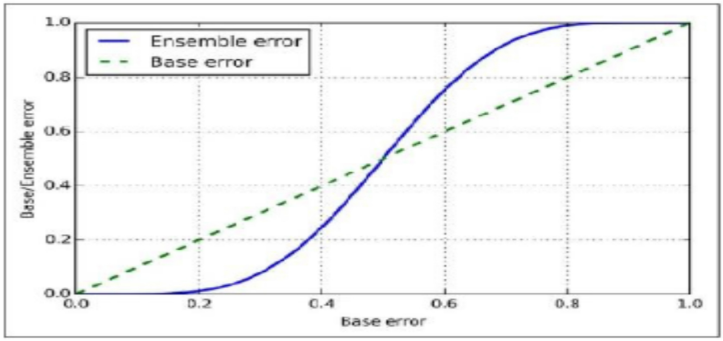
... return sum (probability)

```
>>> ensemble_error(n_classifier = 11, error = 0.25)
0.034327507019042969
```

Let's write some code to calculate the rate for visualization of different errors the relationship between ensemble and base errors in a line graph:

```
>>> import numpy as np
>>> error_Range = no.arrange (0.0, 1.01, 0.01)
>>> en_error = [en_er (n_classifier = 11, er = er)
... in er in er_rang]
>>> Import matplotlib.pyplot as plt
>>> plt.plot (er_range, en_error,
... label = ' ensemble error',
... linewidth = 2)
>>> plt.plot (er_range, er_range,
... ls = '--', label = 'B_ er',
... linewidth = 2)
>>> plt.xlabel ('B_ er')
>>> plt.label ('B / En_er')
>>> plt.legend (loc = 'top left')
>>> plt.grid ()
>>> plt.show ()
```

As we can see in the following plot, the error probability of an ensemble is usually better than the error of an individual base classifier as long as the base classifiers perform better than random predictions ($\varepsilon < 0.5$). You should consider that the y-axis shows the base error as well as the ensemble error (continuous line):



Implementing simple majority classifier

As we saw in the introduction to merge learning in the last section, we will work using a warm-up training and then create a simple classifier for a majority voting in Python programming. As you can see, the next algorithm will work on multiclass settings by plurality voting; you will use the term majority voting for simplicity as common in most articles.

In the following program, we will develop and also join different classification programs associated with individual weights for confidence. Our goal is to create a strong meta-classifier that balances out the individual classifiers' in a particular dataset. In

more precise mathematics onditions, we can write the weighted majority vote.

To translate the idea of a weighted majority vote into Python code, we can

Use NumPy friendly Argmax and bincount functions:

> > > import numpy as np

> > > np.Argemax (np.bincount ([0, 0, 1]),

... weight = [0.2, 0.2, 0.6]))

.

Here, pij is the predicted probability of jth classifier for class label i. To

Continue with our previous example, let's assume we have a binary

Classification problem with the connection of class labels i ∈ {} 0,1 and an ensemble of 3 classifiers Cj (J ∈ {} 1,2,3). Let us assume that Classifier Cj returns

The following class membership probabilities for a specific sample x:

C1 () x → [ ] 0.9,0.1, C2 () x → [ ] 0.8,0.2, C3 () x → [ ] 0.4,0.6

To implement a weighted majority vote based on class probabilities, we can also use NumPy using numpy.average and np.argmax:

> > > ex = NP array ([[0.9, 0.1],

... [0.8, 0.2],

... [0.4, 0.6]])

> > > P = np.average (ex, axis = 0, weight = [0.2, 0.2, 0.6])

> > > p

Array ([0.58, 0.42])

> > > np.argemax (P)

0

Putting everything together, let's now apply a majority vote classifier to it

Python:

from Sklearn.base import classifiermixin

```
from sklearn.pre_processing import from label_En

from sklearn.ext import six

from sklearn.ba import clone

from sklearn.pipline import _name_estimators

Import numpi as np

Import operator

Class MVClassifier (BaseEstimator,

ClassifirMixin):

"" "A majority vote ensemble classifier

Parameters

----------

cl: array-like, shape = [n_classifiers]

Different classifiers for the ensemble votes: str, {'cl_label', 'prob'}

Default: 'cl_label'

The 'cl_label' prediction is based on the argmax of the class labels. Elif 'Probe', The arg of the total probs is used to predict the class label (recommended for calibrated classifier).

W: arr-like, s = [n_cl]

Optional, default: None

If a list of `int or float values is provided, the classifiers are weighed by the " "

def _init _ (s, cl,

v = 'cl_label', w = none):

s.cl = cl

s.name_cl = {key: value for

Key, value in

_name_estimators (cl)}

s.v = V

s.w = w

def fit_CL (s, X, y):
```

"" "Fit_cl.

Parameters

X: (array-like, sparse matrix},

s = [n_samples, n_features]

Matrix of training samples.

y: arr_like, sh = [n_samples]

Vector of target class labels.

Returns

---

s: object

"" "

```
# use LabelEncoder to ensure class labels start
# With 0, which is important for np.argmax
# call in s.predict
s.l_ = LabelEncoder()
s.l_fit (y)
s.cl_ = self.lablenc_.classes_
s.cl_ = []
for cl in s.cl:
fit_cl = clone(cl).fit (X,
s.la_.transform(y)
s.cl_.append(fit_cl)
return s
```

I added many comments to the code to better understand the individual parts. However, before implementing the rest of the methods, let's take a quick break and discuss some of the code that may be confusing at first. We use parent classes Base Estimator and ClassifierMixin to get some support functionality for free, including get_params and set_params to set and return the classifier's parameters as well as the score method for calculating the prediction accuracy, respectively. Also, note that we imported six to

create the MajorityVoteclassifier compatible with Python 2.7.

Next, we will add the prediction method of predicting a class label by a majority vote based on the class labels, if we start a new majorityvoteclassifier object with votes = 'classlabel'. Alternatively, we'll be able to get started the ensemble classifier with vote = 'probability' to predict a class label based on class membership probability. In addition, we will add a prediction_proba method to return the average probabilities, which is useful to calculating the Receiver Operator Characteristic Area under the curve (ROC AUC).

```
Def Pre (s, X):
""" Pre-class labels for X.
Parameters
----------
X:{arr- like, spar mat},
Sh = [n_ samples, n_ features]
Matrix of training samples.
Returns
----------
ma_v: arr-like, sh = [n_samples]
Predicted class labels.
"""
If se.v = = 'probability':
ma_v = np.argmax (spredict_prob(X),
Axis = 1)
else: #'cl_label' v
predictions = np.asarr ([cl.predict (X))
For cl in
s.cl _]). T.
ma_v = np.ap_al_ax (
Lambda X:
np.argmax (np.bincount (x, weight = s.w)),
Axis = 1,
```

```
arr = forecasts)
ma_v = s.l_.inverse_transform(ma_v)
return ma_v
def predict_proba(self, X):
"""prediction for X.
parameters
----------
X: {arr-like, sp mat},
sh = [n_samples, n_features]
Training vectors, where n_sample is the number of samples and n_samples
Number of features.
Returns
----------
av_prob: like array,
sh = [n_samples, n_classes]
Average weight probability for each class per sample.
"""
Probes = np.asarr ([cl predict_probe(x)
for cl in s.cl_])
av_prob = np.av (probes,
axis = 0, weight = s.w)
return av_prob
def get_ps (self, deep= true):
"""Get categorized parameter names for Gridsearch"""
If not deep:
return super (MVC,
self) .gat_ps (deep=False)
else:
ou = s.n_cl.copy ()
```

```
For n, step in\
Six.iteritems (s.n_cl):
for k, value in six. iteritems (
step.get_ps (deep = true)):
ou ['% s _% s'% (n, k)] = value
return ou
```

Combining different algorithms for classification with majority vote

Now, it's almost time to put in the MVC we used earlier in action. You should first create a dataset on which you can test. We are already familiar with the techniques for loading datasets from CSV files, We will take a shortcut and load the Iris dataset from scikit-learn's dataset module.

Also, we will choose only two features, the width of the sepals and the length of the petal make the classification task a bit difficult. Although our multiclass classifier or MVC normalizes to multiclass problems, we will classify only flower samples from two classes, Ir-versicolor and Ir-Virginica to calculate ROC AUC. The code is as follows:

```
> > > Import sklearn as sk
> > > Import sklearn.cross_validation as CV
> > > Ir = datasets.load_ir()
> > > X, y = ir.data [50:, [1, 2]], ir.target [50:]
> > > le = label Encoder()
> > > y = le.fit_transform(y)
```

Next we divide the Iris samples into 50 percent training and 50 percent test data:

```
> > > x_train, x_test, y_train, y_test =\
... train_test_split (X, y,
... test_size = 0.5,
... random_state = 1)
```

Utilizing the training dataset, we now train three different classifiers a logistic regression classifier, a decision tree classifier, and a K-nearest neighbors classifier and study their various performance through a 10 cross-validation

Before we merge them into ensemble one:

Import the following

sklearn.cross_validation

sklearn.linear_model

sklearn.tree

sklearn.pipline

Pipeline

numpy as np

```
>>> clf1 = logistic regression(penalty = 'l2' ',
... C = 0.001,
... random_state =0)
>>> clf2 = DTCL (max_depth = 1,
... criterion = 'entropy',
... random_state = 0)
>>> cl = KNC (n_nb = 1,
... p = 2,
... met = 'minsk')
>>> pipe1 = pipeline ([['sc', StandardScaler()],
... ['clf', clf1]])
>>> pipe3 = pipeline ([['SC', standardscaler()],
... ['clf', clf3]])
>>> clf_labels = ['Logistics Regression', 'Decision Tree', 'KNN']
>>> Print ('10-fold cross validation:\n ')
>>> For clf, label in zip ([pipe1, clf2, pipe3], clf_labels):
... SC = CrossVSc (estimator = clf,
>>> X = X_train,
>>> y = y_train,
>>> cv = 10,
>>> scoring='roc_auc')
```

>>> Print ("ROC AUC:% 0.2f (+ / -%0.2f) [% s]"

...% (scores.mean(), scores.std(), label)

As shown in the following snippet, the output we receive shows that the predictive performance of an individual classifier is almost same:

10-fold cross validation:

ROC AUC: 0.92 (+/- 0.20) [Logistic regression]

ROC AUC: 0.92 (+/- 0.15) [Decision Tree]

ROC AUC: 0.93 (+/- 0.10) [KNN]

You may be wondering why we have trained the logistics regression and K-nearest neighbors classifier as part of a pipeline. The reason here is, as we said, Logistic regression and K-near neighbors algorithms (using Euclidean distance metric) is not scale-invariant unlike with decision trees. Though, the Ir benefits are all measured on an equal scale; It is a good habit to work with standard features.

Now, let's proceed to the more exciting part and connect the individual classifiers for majority rule voting in our M_V_C:

>>> mv_cl = M_V_C (

... cl = [Pipe 1, clf2, pipe 3])

>>> cl_labels + = ['Majority voting']

>>> all_cl = [pipe1, clf2, pipe3, mv_clf]

>>> For cl, label in zip (all_clf, clf_labels):

... sc = cross_val_score(est=cl,

... X = X_train,

... y = y_train,

... cv = 10,

... scoring = 'roc_auc')

...% (scores.mian (), scores.study (), label)

R_AUC: 0.92 (+/- 0.20) [Logistic regression]

R_AUC: 0.92 (+/- 0.15) [D_T]

R_AUC: 0.93 (+/- 0.10) [KNN]

R_AUC: 0.97 (+/- 0.10) [majority vote]

In addition, the output of the MajorityVotingClassifier has substantially improved over the classifiers in 10-fold cross-validation evaluation.

Classifier

In this section, you are calculating the R_C curve from the test set to check if MV_classifier generalizes well to unseen data. We must remember that testing set will not be used for model selection; The only goal is a report an estimation of the accuracy of the classifier system. Let's take a look at imports matrix.

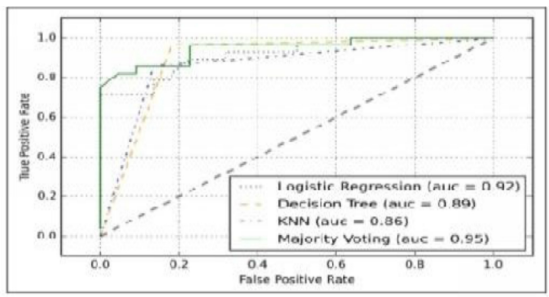
```
import roc_curve from sklearn metrics import auc

cls = ['black', 'orange', 'blue', 'green']

ls = [':', '-', '-.', '-']

For CL, label, cl, 1\

... in zip (all_cl,, cl_labels, cls, ls):

... y_pred = clf.fit (X_train,

... y train) .product_proba (X_test) [:, 1]

... fpr, tpr, thresholds = rc_curve(y_t = y_tes,

... y_sc = y_pr)

... rc_auc = ac (x = fpr, y = tpr)

... plt.plot (fpr, tpr,

... color = clr,

... linestyle = ls,

... la = '% s (ac =% 0.2 f)'% (la, rc_a))

> > > plt.lg (lc = 'lower right')

> > > plt.plot ([0, 1], [0, 1],

... linestyle = '-_',

... color = 'gray',

... linewidth = 2)

> > > plt.xlim ([- 0.1, 1.1])

> > > plt.ylim ([- 0.1, 1.1])

> > > plt.grid ()

> > > plt.xlb ('False positive rate')
```

> > > plt.ylb ('True Positive Rate')

> > > plt.show ()

As we can see in the resulting ROC, the ensemble classifier also performs well

On the test set (ROC AUC = 0.95), while K-near neighbors classifier seems to be over-fitting the training (training ROC AUC = 0.93, test ROC AUC = 0.86):


True Positive Rate
False Positive Rate
Logistic Regression (auc = 0.92)
Decision Tree (auc = 0.89)
KNN (auc = 0.86)
Majority Voting (auc = 0.95)

You select only two features for classification functions. It will be interesting to show what the decision field of ensemble classifier looks like. However it is not necessary to standardize the training features before the model to fit because our logistics regression and K-nearest neighbors pipelines will take care of this automatically, you will set up training set so that the decision regions of the decision tree will be on the same scale for visual purposes.

Let's dive in:

> > > sc = SS ()

X_tra_std = sc.fit_transform(X_train)

From itertools imported product

```
x_mi = X_tra_std [:, 0] .m () - 1
x_ma = X_tra_std [:, 0] .ma () + 1
y_mi = X_tra_std [:, 1] .mi() - 1
y_ma = X_tra_std [:, 1] .ma () + 1
xx, yy = np.meshgrid(np.arange (x_min, x_max, 0.1),
... np arange (y_mi, y_ma, 0.1)
f, axarr = plt.subplots (nrrows = 2, ncols = 2,
Sharex = 'col',
sharey = 'row',
figze = (7, 5)
for ix, cl, tt in zip (product ([0, 1], [0, 1])),
all_cl, cl_lb):
... cl.fit (X_tra_std, y_tra)
... z = cl.f.predict (np.c_ [xx.ravel (), yy.ravel ()])
... z = Z.res (xx.shape)
... character [IDX [0], IDX [1]]. Contu (_xxx, _ ye, z, alf = 0.3)
... character [idx [0], idx [1]]. Scatter (X_tra_std [y_tra == 0, 0],
... X_tra_std [y_tra == 0, 1],
... C = 'Blue',
... score = '^',
... s = 50)
... character [idx [0], idx [1]]. Skatt (X_tra_std [y_tra == 1, 0],
... X_tra_std [y_tra == 1, 1],
... c = 'red',
... marker = 'O',
... s = 50)
... character [idx [0], idx [1]]. set_title (tt)
>>> plt.text (-3.5, -4.5,
... z = 'slay wide [standard]',
```

... yes = 'center', w = = 'center', ftsize = 12)

> > > plt.text (-10.5, 4.5,

... z = 'p_length [standard]',

... yes = 'center', va = 'center',

... f_size = 13, rotation = 90)

> > > plt.show ()

Interestingly, but as expected, the areas of decision of the class included

The decision of the individual classified class seems to be a hybrid of regions. At

At first glance, the majority vote seems like a boundary decision

K-nearest neighbor classifier boundary. However, we can see that it is

Like the stump of the decision tree, the orthogonal for the y axis for the sapphire width-1

Before you learn how to tune individual classifier parameters for configuration

Classification, let's call the get_ps method to find the essential idea of how we can

Access the individual parameters inside the grid search object budget:

> > > mv_clf.get_params ()

decision 'decantentriclassifier': decantentriclassifier (class_weight = nothing,

criterion = 'entropy', maximum_depth = 1,

Max_tails = nothing, max_leaf_nodes = nothing, minute_samples_

Leaf = 1,

Minute_samples_split = 2, minute_weat_fraction_leaf = 0.0,

random_state = 0, splitter = 'best'),

decisiontreeclassifier_class_weight': none,

decisiontreeclassifier_'s class': 'entropy',

[...]

decisiontreeclassifier_random_state': 0,

decisiontreeclassifier_splitter: 'best',

'Pipeline-1': pipeline (steps = [('sc', StandardScaler (copy= true, with_

mean = True, with_std = True)), ('clf', Logistic Regression(C = 0.001, Class_weight=None,

dual = False, fit_incept = true,

intercept_scaling = 1, max_ iter = 100, multi_class = 'ovr',

penalty = 'l2', random_state = 0, solver = 'libliner',

toll = 0.0001,

verbose = 0),

'Pipeline-1_ clf_C': 0.001,

'Pipeline-1_ clf_class_weight': None,

'Pipeline-1_ clf_dual':False,

[...]

'pipeline-1_sc_with_std': True

'pipeline-2': Pipeline(steps=[('sc', StandardScaler(copy=True, with_

mean=True, with_std=True)), ('clf', KNeighborsClassifier(algorithm='au

to', leaf_size=30, metric='minkowski',

metric_params=None, n_neighbors=1, p=2,

w='uniform'))]),

'p-2_cl": KNC(algorithm='auto', leaf_

size=30, met='miski',

met_ps=None, n_neighbors=1, p=2,

w='uniform'),

'p-2_cl_algorithm': 'auto',

[...]

'p-2_sc_with_std': T}

Depending on the values returned by the get_ps method, you now know how to access the individual classifier's attributes. Let's work

with the inverse regularization parameter C of the logistic regression classifier and the decision tree depth via a grid search for demonstration purposes. Let's take a look at:

```
>>> from sklearn.grid_search import GdSearchCV
>>> params = {'dtreecl__max_depth': [0.1, .02],
... 'p-1__clf__C': [0.001, 0.1, 100.0]}
>>> gd = GdSearchCV(estimator=mv_cl,
... param_grid=params,
... cv=10,
... scoring='roc_auc')
>>> gd.fit(X_tra, y_tra)
```

After the grid search has completed, we can print the different hyper parameter value combinations and the average R_C AC scores computed through 10-fold cross-validation. The code is as follows:

```
>>> for params, mean_sc, scores in grid.grid_sc_:... print("%0.3f +/-%0.2f %r"
... % (mean_sc, sc.std() / 2, params))
0.967 +/-0.05 {'p-1__cl__C': 0.001, 'dtreeclassifier__ma_depth': 1}
0.967 +/-0.05 {'p-1__cl__C': 0.1, 'dtreeclassifier__ma_depth': 1}
1.000 +/-0.00 {'p-1__cl__C': 100.0, 'dtreeclassifier__ma_depth': 1}
0.967 +/-0.05 {'p-1__cl__C': 0.001, 'dtreeclassifier__ma_depth': 2}
0.967 +/-0.05 {'p-1__cl__C': 0.1, 'dtreeclassifier__ma_depth': 2}
1.000 +/-0.00 {'p-1__cl__C': 100.0, 'dtreeclassifier__ma_depth': 2}
>>> print('Best parameters: %s' % gd.best_ps_)
Best parameters: {'p1__cl__C': 100.0,
'dtreeclassifier__ma_depth': 1}
>>> print('Accuracy: %.2f' % gd.best_sc_)
Accuracy: 1.00
```